Universidad de las Américas Puebla

Escuela de Ingeniería

Departamento de Computación, Electrónica y Mecatrónica

“Sistema de mapeo tridimensional en línea basado en código abierto: Aplicación en regiones arqueológicas”

Tesis que, para completar los requisitos del Programa de Honores, presenta el estudiante

Johan Pablo Loredo Pineda

Jurado de examen profesional:

Dr. Oleg Starostenko Basarab
Presidente

Dr. César Martínez Torres
Asesor

Dr. Juan Carlos Galán Hernández
Secretario

Dr. José Martínez Carranza
Co-asesor

Agradecimientos

Se agradece al Dr. José Martínez Carranza, así como a los miembros del equipo Quetzal C++ por su amplio apoyo a la realización de este proyecto a través del Instituto Nacional de Astrofísica, Óptica y Electrónica.

Se agradece al Dr. Shigeru Kabata por su accesibilidad y disposición para permitir la realización de experimentos de este sistema dentro del marco de su investigación sobre la zona arqueológica de Tlalancaleca, Puebla.

Se agradece también al Dr. Gregorio Martínez y a Maria Lisset Abundis de la Mora por su apoyo en el desarrollo del proyecto.

Finalmente, se agradece a las siguientes instituciones: CONACYT y INEGI en México; y a la Royal Society y el Fondo Newton en el Reino Unido, ya que en este trabajo se utilizó equipo adquirido a través del financiamiento otorgado por el Fondo Sectorial CONACYT-INEGI, proyecto no. 268528, y el proyecto RAFAGA con número NA140454.

Índice

Sección	Página
Índice de contenido	3
Resumen	5
Introducción	5
Capítulo 1: Planteamiento del problema	7
1.1: Objetivos	9
1.1.1: Objetivo general	9
1.1.2: Objetivos específicos	9
1.1.3: Limitaciones	10
Capítulo 2: Antecedentes	11
2.1 Mapeo asistido por computadora	11
2.2: Imagen aérea	12
2.3: Reconstrucción tridimensional	14
2.4: Fotogrametría de estructura del movimiento	16
Capítulo 3: Desarrollo del sistema	19
3.1: Herramientas de código abierto	20
3.1.1: Open Drone Map	20
3.1.2: Open Source Computer Vision Library	22
3.1.3: Point Cloud Library	23
3.1.4: Meshlab	24
3.2: Algoritmo	25
3.2.1: Filtrado de imágenes	25
3.2.2: Mapeo	31
3.2.3: Visualización	33

Capítulo 4: Open Drone Map contra Agisoft 35

Capítulo 5: Desarrollo experimental y resultados 43

Capítulo 6: Conclusiones 52

Capítulo 7: Trabajo futuro 54

Bibliografía 55

Resumen

El presente documento muestra el desarrollo de un sistema capaz de realizar un procedimiento de filtrado de imágenes capturadas a través de un vehículo aéreo no tripulado, a partir de las cuales se realiza la generación de una malla tridimensional texturizada capaz de proveer información relevante para el estudio remoto de zonas arqueológicas.

El sistema fue desarrollado utilizando únicamente herramientas de código abierto, y los resultados que entrega fueron comparados contra un software de paga enfocado al mapeo tridimensional.

Introducción

En tiempos recientes se ha observado que el área tecnológica que se refiere a los vehículos aéreos no tripulados (UAV's por sus siglas en inglés) avanza con gran rapidez, lo cual permite que estos sean utilizados para diversas actividades de carácter profesional o recreativo. El uso de los UAV's como una herramienta a través de la cual obtener imágenes aéreas no es algo nuevo, pues, en sus inicios por parte de la milicia en los años 60, estos eran utilizados para captar imágenes de bases enemigas sin tener que comprometer vidas humanas; actualmente, son utilizados en una muy amplia gama de aplicaciones, que van desde la industria del cine, fotografía y deporte, hasta su uso como herramientas de investigación en distintas áreas de la academia.

En este caso, la versatilidad que poseen dada la facilidad de su pilotaje, aunada a su reducido tamaño y relativo bajo costo, permite que sean explotados para realizar mapeos no invasivos de zonas arqueológicas, de modo que estas regiones puedan ser estudiadas detalladamente con la ventaja de reducir al mínimo el riesgo de causar daños estructurales derivados de la exploración de tipo invasiva como el mapeo a través de escáner láser terrestre.

Esto puede ser logrado de distintas maneras, como el uso de herramientas de paga

desarrolladas ya sea para la generación de mapas tridimensionales, o para el desarrollo de mosaicos de imágenes aéreas, sin embargo, en este caso se optó por explotar diversas herramientas de código abierto existentes que permiten a los desarrolladores cumplir una determinada tarea o proceso computacional de una manera más sencilla, pues estas herramientas, por lo general, se encargan de proporcionar una solución práctica a algoritmos tanto sencillos como de alta complejidad. Además de que existen sistemas de código abierto capaces de realizar algunas de las tareas específicas con las que cuentan herramientas de software pagadas, llegando incluso a obtener resultados mejores, en algunos casos, siempre y cuando estos sean implementados de la manera adecuada.

Es por esto que se introduce la idea de desarrollar un sistema completo capaz de producir mapas tridimensionales de una zona arqueológica en la que este se encargue tanto de la obtención de imágenes aéreas, como de la creación del modelo tridimensional computado a partir de los datos obtenidos, permitiendo así que la atención del usuario sea enfocada únicamente a pilotear la herramienta de obtención de imagen (típicamente un dron) sobre un área determinada de interés, incrementando el nivel de eficiencia durante la jornada de mapeo, de modo que la cantidad de datos obtenidos en una sola jornada incremente.

Para desarrollar un sistema con tales características se detectaron las herramientas útiles para este propósito, siendo estas principalmente enfocadas al mapeo y al procesamiento de imágenes. Además de lo cual fue necesario el planteamiento y la implementación de un algoritmo capaz de unir estos procesos para ejecutarlos como uno mismo, de ser necesario; para lo que se requirió también un sistema operativo que permitiera la comunicación en tiempo real del procesamiento con la transmisión de datos de video por parte del vehículo aéreo.

Capítulo 1: Planteamiento del problema

Es innegable que el software de paga desarrollado por diversas empresas o centros de investigación cuenta con un excelente nivel de calidad, pues este ha de ser creado teniendo en cuenta que su precio debe ser al menos proporcional a la funcionalidad que ofrece. Sin embargo, estos pueden resultar excesivamente costosos para la funcionalidad específica que un grupo o individuo desea darles, pues en algunos casos la característica aislada que se desea utilizar es únicamente encontrada dentro de paquetes de software que incluyen diversas herramientas.

Actualmente, los avances en el área de programación han permitido que librerías y repositorios de diversos ámbitos para resolver todo tipo de algoritmos sean desarrollados como código abierto. Esto significa que el acceso a ellos es libre y gratuito para la comunidad, de modo que pueden ser utilizados por cualquier persona interesada en el desarrollo de software cuyo código así lo requiera; siendo que, en algunos, casos los autores solicitan ser citados si su trabajo está siendo utilizado para contribuir a un trabajo de desarrollo o investigación científica. En caso de que el algoritmo de código abierto quiera ser utilizado para un fin comercial, los autores tienden a solicitar ser contactados para discutir los términos de uso de sus librerías.

En este caso, el problema se centra en el uso de imágenes aéreas como un medio no invasivo para la obtención de mapas tridimensionales de zonas arqueológicas, es decir, obtener una imagen virtual que es posible explorar de manera remota sin comprometer la integridad de las estructuras estudiadas. Además, el sistema no está limitado a su uso con imágenes aéreas, sino que puede ser utilizado para procesar imágenes terrestres para obtener modelos no solo de estructuras, sino también de piezas frágiles cuya inspección manual podría resultar en daños significativos hacia esta.

Para ello, se utilizó como base el trabajo del Dr. Shigeru Kabata, quien trabaja en una

investigación sobre la zona arqueológica de Tlalancaleca, la cual se ubica a 45 minutos de la capital del estado de Puebla, y encuentra especialmente útil el uso de mapas tridimensionales texturizados para el estudio de la zona, los cuales obtiene al generar una base de datos que consiste en una serie de imágenes aéreas sobre las regiones que desea observar detalladamente. Para llevar a cabo el proceso de mapeo tridimensional en zonas arqueológicas el Dr. Kabata utiliza el software de Agisoft (Agisoft Photoscan, 2018), desarrollado por la compañía del mismo nombre fundada en 2006, que se enfoca en la tecnología de visión computacional a partir de algoritmos de procesado de imágenes con técnicas de fotogrametría digital. Para su uso, la base de datos generada por los investigadores debe contar con imágenes, ya sean aéreas verticales u horizontales, o captadas desde la tierra, dependiendo de si se está generando el modelo de una pieza o estructura de reducidas dimensiones, o una región completa.

Una de las más grandes dificultades presentadas al realizar el trabajo de investigación del Dr. Kabata es que la recopilación de imágenes se realiza de manera manual, lo que significa que el operador del UAV debe reubicar el vehículo hasta determinar subjetivamente el momento adecuado en el cual tomar cada cuadro, haciendo una estimación mental del porcentaje de semejanza entre la última imagen capturada y la imagen actual, lo cual implica un alto costo en cuanto al tiempo de trabajo y, dado que la ubicación de la zona podría considerarse hasta cierto punto remota, esto resulta problemático si se toma en cuenta que el tiempo disponible para la obtención de imágenes en cada jornada estará limitado a la cantidad de baterías de dron completamente cargadas disponibles, multiplicado por el tiempo de vuelo que estas ofrecen.

Finalmente, el método de obtención de mapas utilizado actualmente, impide a los investigadores saber si es necesario capturar más imágenes de cierto ángulo de la zona, o si es mejor repetir completamente el vuelo, pues una vez que las imágenes han sido obtenidas, el

procesamiento de las mismas para generar una nube de puntos, y posteriormente una malla texturizada sobre la misma, es llevado a cabo después de retirarse de la región; por lo que cualquier corrección necesaria, detectada como resultado del mapeo, deberá postergarse hasta el siguiente viaje a la zona arqueológica.

1.1: Objetivos

A continuación se describe el objetivo de este proyecto, así como los objetivos específicos que permitirán cumplir este objetivo y solucionar la problemática planteada.

1.1.1: Objetivo general

Desarrollar un sistema basado en código abierto de mapeo en línea a partir de imágenes aéreas provenientes de la señal de video de un dron mientras este se encuentra en vuelo, el cual sea capaz de producir resultados comparables con los mapas texturizados entregados por un sistema de mapeo de licencia de paga, y que realice la captura de datos de una manera más eficiente, disminuyendo el tiempo de procesamiento para la generación del mapa.

1.1.2: Objetivos específicos

Seleccionar herramientas de código abierto para el procesamiento de imágenes a partir de las que se genera un mapa tridimensional.

Realizar un sistema que filtre imágenes bajo un umbral de coincidencia, definido en código, a partir de la señal de video que produce un dron durante su vuelo.

Implementar un sistema que inicie el proceso de mapeo cuando se cumplan condiciones específicas.

Desarrollar un visualizador de nubes de puntos con la posibilidad de ser abierto desde la terminal para observar los resultados mientras el resto de procesos siguen activos.

Generar un código capaz de unir los procesos de los objetivos previos de modo que estos se ejecuten como un mismo proceso de mapeo.

Comparar los resultados que ofrece el sistema contra el proceso de mapeo llevado a cabo por el Dr. Shigeru Kabata sobre la zona arqueológica de Tlalancaleca.

1.1.3: Limitaciones

Todas las herramientas de procesamiento de datos utilizadas para el desarrollo de este proyecto fueron obtenidas como código abierto y usadas de acuerdo a las licencias con las que estas fueron liberadas.

El factor determinante para la selección de la mayoría de las herramientas descritas fue su compatibilidad con la terminal de Ubuntu.

El proyecto se limita únicamente a la entrega de un mapa tridimensional texturizado, sin ningún tipo de post procesamiento a partir de dicho punto.

El sistema desarrollado funciona únicamente a través de ROS, por lo que es necesario utilizar un UAV compatible con este sistema operativo para su funcionamiento en línea; es posible utilizar drones no compatibles con ROS, pero esto implica que todo el sistema sea utilizado después de que el vuelo ha finalizado, pues en este caso se utiliza un script que publica un tópico a partir de un video existente, almacenado en una ubicación conocida.

Capítulo 2: Antecedentes

2.1: Mapeo asistido por computadora

Los orígenes de los mapeos asistidos por computadora se observaron desde la década de 1970, en la que se introdujeron por primera vez elementos computacionales a la producción de mapas topográficos, entre los cuales se destacan elementos de hardware como terminales alfanuméricas, interfaces electrónicas, y mesas de trazado electrónico, así como elementos de software como programas de fotogrametría y cartografía enfocados a ayudar al operador a realizar operaciones repetitivas (Ramirez, 1988c). A partir de este primer acercamiento tecnológico sobre esta área, se comenzó un largo camino de optimización al proceso de la obtención de diferentes tipos de mapas en diferentes escalas; se crearon nuevas herramientas tanto de software como de hardware para apoyar con esta tarea, como lo son escáneres láser, sensores ultrasónicos o infrarrojos; o bien el desarrollo de nuevos algoritmos que permiten que muchos procesos de mapeo sean realizados automáticamente permitiendo que la interacción del operador sea mínima a lo largo del proceso.

En cuanto al mapeo de regiones arqueológicas usando asistencia por computadora en México, se puede hablar del Proyecto de Mapeo de Palenque (PMP) que data del año 1998, en el que "se registraron casi 10,000 puntos para topografía y arquitectura. La precisión del mapa, verificada al cerrar polígonos de estaciones EDM (Electronic Distance Meters) Topcon, se mantuvo debajo de ±10 cm de error" (Barnhart, 2004). Según lo descrito en su metodología se incluye el uso del instrumento de medición por medio de reflejos de luz Topcon GTS2R, con el que se generaron mediciones registradas como puntos tridimensionales que fueron manipulados utilizando un antiguo software de mapeo llamado Foresight, cuyos datos fueron posteriormente modificados utilizando AutoCAD, dando como resultado el mapa mostrado en la Figura 1.

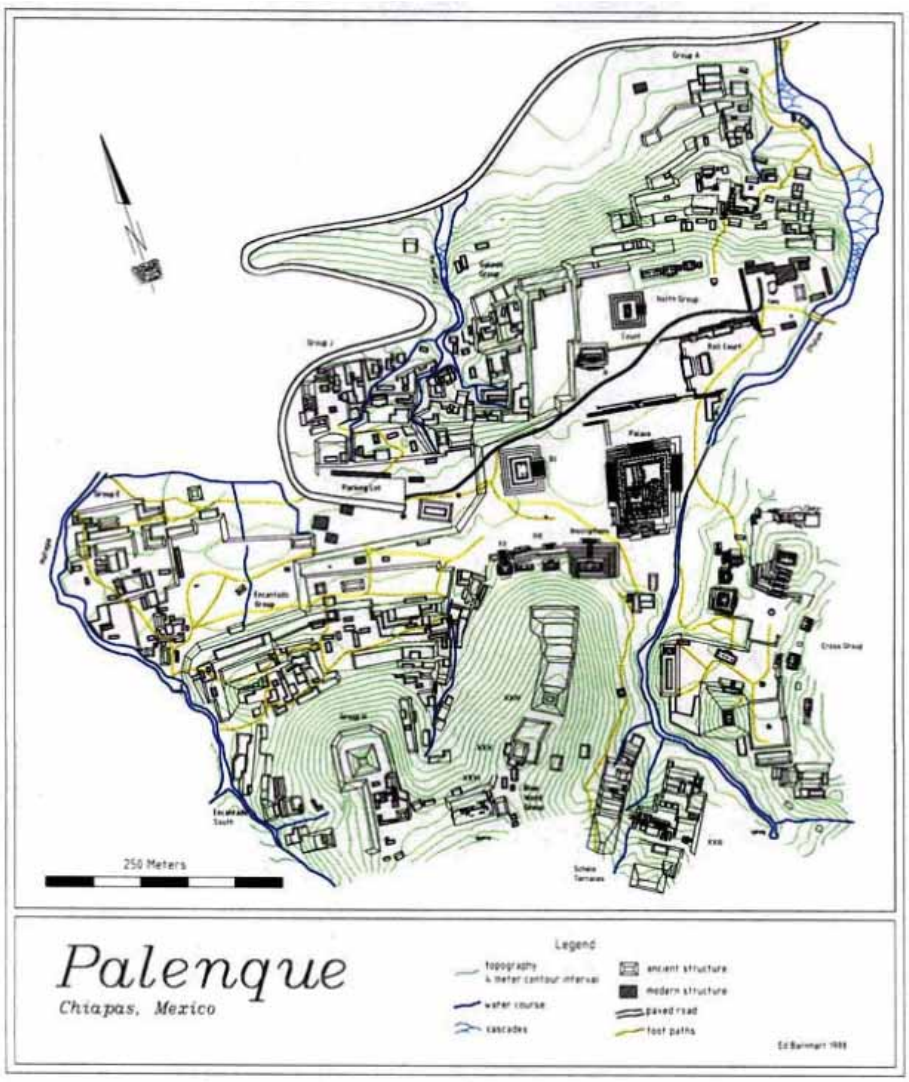

Figura 1: Mapa de Palenque, Chiapas, México, generado a través del PMP. Uno de los primeros mapas desarrollados utilizando asistencia de computadora en México.

2.2: Imagen aérea

Siendo actualmente una de las herramientas vitales para disciplinas tales como la arqueología de paisaje (el estudio de cómo el ser humano ha utilizado el paisaje en el pasado), geografía humana (el estudio del uso moderno del paisaje), o la ciencia climatológica (para monitorear el comportamiento del clima en determinadas zonas a través de los cuerpos de agua o hielo), la fotografía aérea fue originada en 1858 por el fotógrafo Gaspard Felix Tournachon, también

conocido como Nadar, quien tomó fotografías de París usando un globo atado a 1600 pies de altura. Para ser posteriormente mejorada por James Wallace Black al tomar fotografías desde un globo aerostático en 1860. No fue sino hasta la Primera Guerra Mundial que se encontraron aplicaciones militares para la fotografía aérea, pues estas eran capaces de brindar una gran cantidad de información de reconocimiento sobre bases enemigas o en la distribución de tropas en campo abierto, así como la localización de obstáculos naturales que pudieran complicar una misión o para encontrar la ruta más eficaz y menos comprometedora para llevarla a cabo.

La obtención de imágenes aéreas tuvo un avance prácticamente proporcional al avance de la fotografía en sí, pues al obtener cámaras más ligeras y de mecanismos más sencillos, se vuelve menos complejo el proceso de montarlas sobre vehículos aéreos no tripulados. De esta manera, la fotografía aérea logró abrirse camino como una de las más importantes herramientas complementarias para el mapeo, pues, tal como era el objetivo de Nadar, esto resulta mucho más efectivo, y menos costoso temporalmente hablando, que los métodos de mapeo terrestre utilizados previamente, pues una sola imagen del terreno tomada desde las alturas es capaz de brindar mucha más información que un conjunto de datos obtenidos utilizando escáneres láser en mucho menos tiempo.

Dependiendo del grado de inclinación con el que las imágenes aéreas son captadas, estas pueden ser clasificadas como imágenes verticales, en las que el foco es apuntado con un ángulo no mayor a 3° respecto del eje vertical con el suelo, lo que significa que las imágenes muestran el terreno que se encuentra directamente debajo del vehículo u objeto desde el que estas son capturadas, este tipo de imágenes es el que se utiliza típicamente para la elaboración de mapas y planos. Por otro lado, se encuentran las imágenes oblicuas, en las que el foco es apuntado con un ángulo mayor a 3° respecto del eje vertical, las cuales son subdivididas en imágenes aéreas bajas (entre 3° y 45° respecto del eje vertical), en las que no se observa el

horizonte; e imágenes aéreas altas o panorámicas (entre 45° y 90°), en las que el horizonte y el cielo son observables, por lo que son normalmente utilizadas para fines más acercados hacia el aspecto artístico o para fines turísticos.

2.3: Reconstrucción tridimensional

Una de las primeras herramientas de reconstrucción de modelos tridimensionales es el escáner láser terrestre (TSL - Terrestrial Laser Scanning), un sistema en el que, como lo indica su nombre, se utiliza una emisión de rayos láser a una frecuencia mínima de 1 kHz para detectar objetos o puntos que perturben la longitud de onda de la luz. Rudolf Staiger (2003) señala en su trabajo "Terrestrial Laser Scanning Technology, Systems and Applications" que este proceso se divide en tres partes principales, las cuales a su vez pueden ser subdivididas, comenzando por la adquisición de datos, en la que se escanea al área de interés es escaneada en una o varias tomas, lo que resulta en una nube de puntos tridimensionales al hacer una unión de las coordenadas de puntos que se obtienen en cada toma. Posteriormente, la nube de puntos generada debe de ser procesada para eliminar perturbaciones, las cuales pueden ser agujeros sin información, o bien puntos de objetos que no existen, los cuales pueden ser causados por agentes móviles en la escena capturada; después de lo cual, el procesamiento puede ser dirigido hacia calcular elementos geométricos de una subsección de la nube de puntos, la intersección de elementos geométricos, la combinación de la información escaneada con imágenes digitales, o simplemente exportación de la nube hacia otros programas para postprocesamiento. Finalmente, se encuentra la etapa de visualización que puede ir desde una proyección de nube de puntos con representación de profundidad, hasta los planos de contorno o modelos 3D.

En lo que respecta a la reconstrucción de modelos tridimensionales usando visión computacional, el doctor Huei-Yung Lin detalla desde el año 2002 una metodología para el

cumplimiento de este objetivo haciendo uso de una cámara monocular montada sobre un riel que ayuda a simular el efecto estéreo, para lo cual se utiliza un motor a pasos que se encarga del movimiento de la cámara sobre un eje con la precisión necesaria para utilizar los parámetros de desplazamiento dentro de la matriz de traslación necesaria para realizar el mapeo. Aunado a esto, el objeto a mapear se encuentra sobre una base rotatoria, la cual es rotada bajo un ángulo determinado para complementar al registro de la pieza en 360°. La solución de Lin para la reconstrucción de modelos tridimensionales, mostrada de forma gráfica en la Figura 2, consta de cuatro partes: Adquisición de datos, registro, integración de superficie, y mapeo de texturas, y resultó en uno de los primeros algoritmos de reconstrucción tridimensional automática con resultados de textura fotorrealista usando una cámara monocular digital.

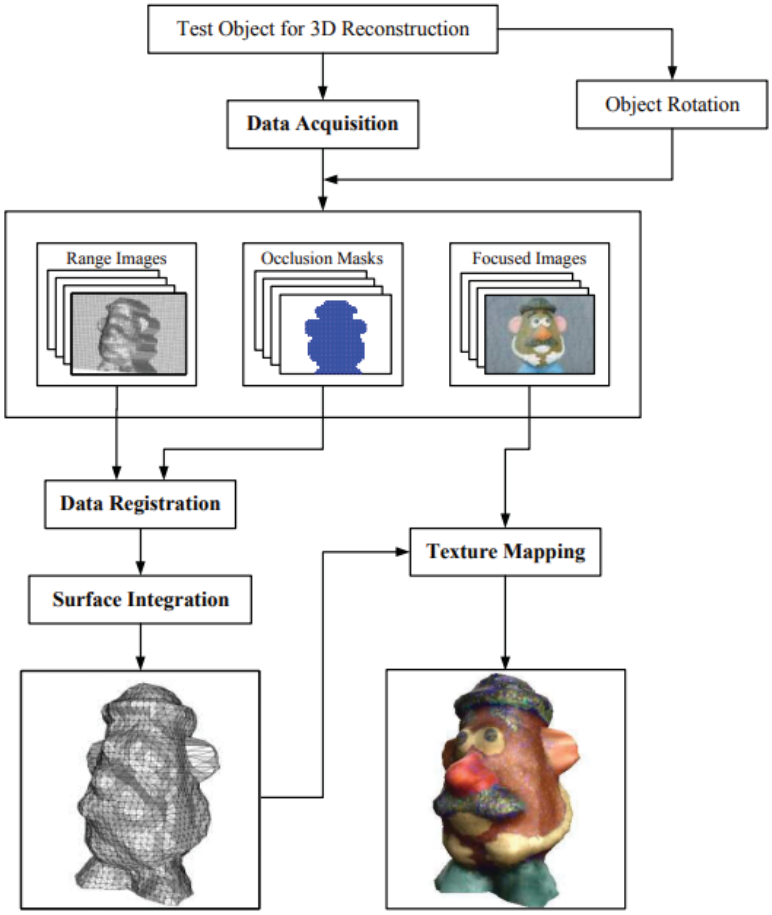

Figura 2: Diagrama de flujo para la reconstrucción completa de un modelo tridimensional (Lin, 2002).

2.4: Fotogrametría de estructura del movimiento

Finalmente, se puede hablar de la fotogrametría de estructura del movimiento (SfM - Structure from Motion), el cual está basado en la fotogrametría tradicional junto con los algoritmos de función de coincidencia (Förstner, 1986) para eliminar la necesidad de comenzar el mapeo introduciendo manualmente puntos de control identificados en las imágenes de entrada y haciendo un proceso conocido como *estimación de pose de la cámara* para determinar la posición de la cámara. En cambio, la SfM no requiere de ningún conocimiento previo a la reconstrucción; la pose de la cámara y la geometría de la escena son

computadas junto con la identificación de características coincidentes en múltiples imágenes. Esto causa que la nube de puntos tridimensional sea generada en un sistema de coordenadas relativo al espacio de la imagen, por lo que debe ser alineado con un eje coordenado del mundo real, lo cual puede ser logrado al referenciar puntos conocidos dentro de la nube (Westoby, Brasington, Glasser, Hambrey y Reynolds, 2012).

La estructura del movimiento parte desde el uso de la visión computacional para seguimiento de puntos a través de secuencias de imágenes de diferentes posiciones, y debe su existencia a diversas innovaciones matemáticas y computacionales a lo largo del tiempo. Las condiciones de coplanaridad, por ejemplo, fueron usadas entre 1950 y 1960 para la triangulación numérica aérea y mapeo a partir de imágenes aéreas (Thompson, 1965). El ajuste de paquetes y relación entre imágenes fue establecido por Brown (1971, 1976), Kenefick et al. (1972), y Granshaw (1980). Sin olvidar la autocalibración de la cámara (Kenefick et al., 1972; Faig y Moniwa, 1973) que permiten modelar y estimar los parámetros internos de la cámara, que permite eliminar cierto grado de distorsión contenido en todas las cámaras fotográficas, y que puede afectar al resultado final de la estructura del movimiento.

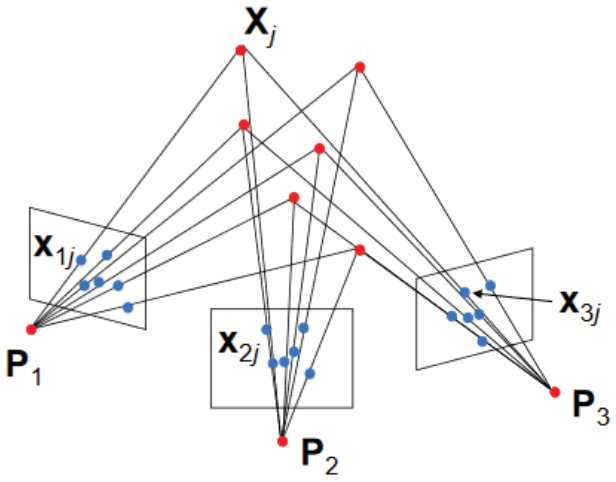

Figura 3: Representación de la obtención de una nube de puntos tridimensional a través de imágenes usando Structure from Motion (Fergus, 2013).

Este procedimiento ha sido optimizado en tiempos recientes, por lo que se han alcanzado algoritmos capaces de realizar mapeos con una mayor precisión y densidad, y un menor costo temporal, en los que se obtiene una nube densa de puntos, lo cual ayuda a mejorar la percepción visual de la escena mapeada, o bien, del objeto escaneado para su análisis. Además, comparado con el TLS, el procedimiento para obtener un mapa tridimensional con puntos coloreados es mucho más sencillo, pues, dado que la triangulación de los puntos se obtiene directamente de los puntos clave encontrados en imágenes, es posible extraer el color del pixel del que proviene cada punto y asignarlo a este junto con la matriz de posición al momento de incluirlo en la nube. Además, al no requerir calibración alguna antes de comenzar el mapeo, es posible comenzar con el proceso desde cualquier momento, lo que permite que el análisis sea iniciado una vez que el vehículo o la cámara se encuentran en una posición y altura favorable para evitar la redundancia de puntos correspondientes al inicio del procesamiento.

A modo de conseguir buenos resultados utilizando este método, una serie de puntos clave han sido propuestos por Micheletti et al. en 2014, incluyendo los siguientes:

- Planear un método de registro o escala fotográfico por adelantado.
- Capturar completamente al sujeto al inicio, y posteriormente los detalles, asegurándose de que la oclusión sea capturada adecuadamente.
- Asegurar la cobertura apropiada. Un principio básico es que todos los puntos del sujeto deben aparecer al menos en tres de las imágenes adquiridas desde diferentes ubicaciones espaciales.
- Escena estática.
- Luz consistente.
- Evitar imágenes sobreexpuestas o subexpuestas.
- Evitar imágenes borrosas - que son normalmente producidas por un lento tiempo de exposición, o movimiento de la cámara.
- Evitar superficies transparentes, reflejantes, u homogéneas.

Capítulo 3: Desarrollo del sistema

En este apartado se hablará sobre las especificaciones de sistema operativo bajo las que el sistema fue desarrollado, así como de las herramientas tomadas en cuenta para el mismo motivo y el funcionamiento del algoritmo que lleva a cabo el procesamiento.

Para comenzar, el sistema operativo utilizado fue Ubuntu 16.04.4 LTS (Xenial Xerus) basado en la serie 4.4 de Linux; esto debido a que otro de los elementos base del proyecto es Robot Operating System (ROS), el cual tiene una mejor compatibilidad con el sistema de Ubuntu, comparado con sistemas de Windows o Mac OS.

En cuanto a la ROS, se utilizó la versión Kinetic Kame para establecer la comunicación del equipo con el dron Parrot Bebop, utilizado en las etapas de experimentación, para lo cual se utilizaron los drivers de bebop_autonomy, desarrollados en el Laboratorio de Autonomía de la universidad de Simon Fraser, los cuales se encargan de definir en formato de código los mensajes que pueden ser enviados hacia el vehículo para su control.

Robot Operating System es un software de código abierto bajo licencia BSD, una licencia de software permisiva con pocas restricciones y que permite el uso de código para casi cualquier tipo de distribución; el cual tiene un enfoque hacia ayudar a los desarrolladores a crear aplicaciones robóticas a través de sus múltiples herramientas como abstracción de hardware, drivers de dispositivos, librerías, visualizadores, administración de paquetes, entre otros. ROS funciona haciendo uso de nodos para establecer comunicación entre los componentes de un sistema robótico, los cuales no deben de estar ubicados necesariamente en el mismo sistema, o incluso la misma arquitectura computacional; utiliza un nodo principal a través del que los demás nodos se comunican, de esta manera, un nodo que desea enviar un mensaje a otro se convierte en un publicador, el cual envía su mensaje hacia el nodo principal bajo un tópico específico, mientras que el nodo que debe recibir el mensaje, convirtiéndose en el suscriptor, lee continuamente al nodo principal hasta encontrar algún mensaje bajo el tópico que le ha

sido asignado.

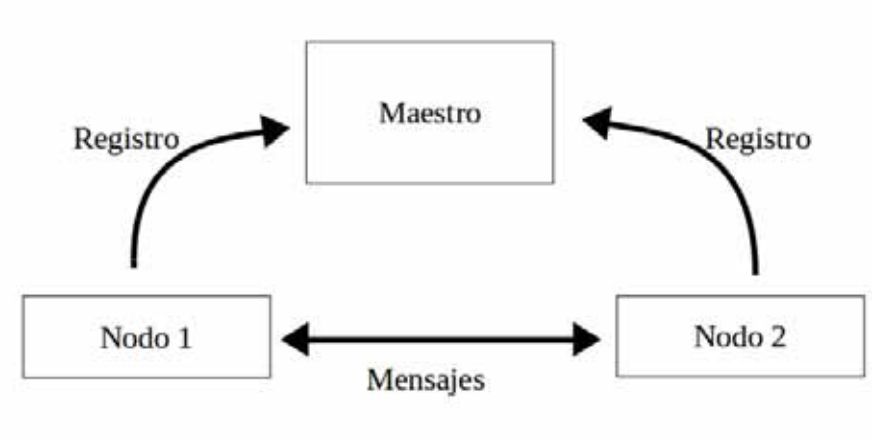

Figura 4: Representación en nivel alto del funcionamiento de ROS.

La interpretación de la Figura 4 implica que cuando un nodo publicador se registra al maestro, éste declara que publicará un tópico específico, del mismo modo, cuando un nodo suscriptor se registra, declara que se suscribirá a un tópico específico.

3.1: Herramientas de código abierto

A continuación se describen las herramientas de desarrollo disponibles como código abierto que fueron utilizadas para el desarrollo de cada etapa de este proyecto.

3.1.1: Open Drone Map

Open Drone Map (ODM) es un kit de herramientas de código abierto para el procesamiento de imágenes aéreas desarrollado por Michael Waechter, Nils Moehrle, y Michael Goesele, capaz de producir productos como nubes de puntos, modelos digitales de superficie, modelos digitales de superficie texturizados, imágenes ortorectificadas, modelos digitales de elevación, entre otros.

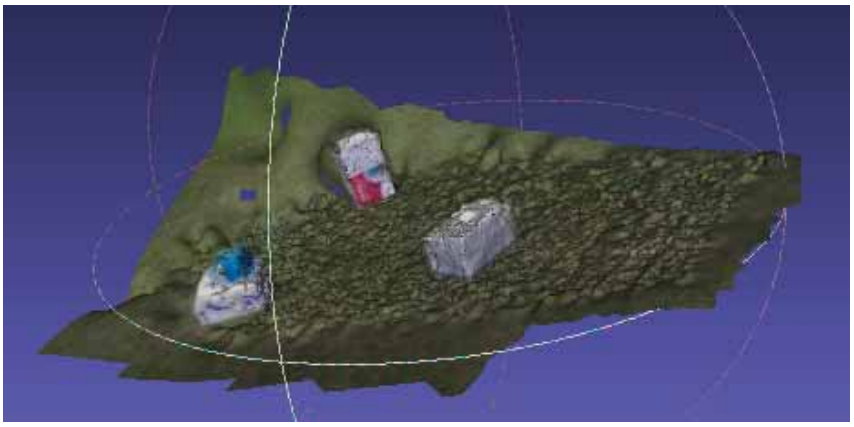

Figura 5: Mapa tridimensional de tres cajas colocadas sobre el pasto generado por Open Drone Map.

Para la generación de nubes de puntos, ODM toma imágenes de un paquete y las procesa de una en una para encontrar puntos clave en cada una de ellas. Después, toma los puntos de cada imagen, denominados descriptores, para buscarlos dentro del resto de imágenes contenidas en la base de datos, lo que le permite computar una triangulación a través de la cual se obtiene una estimación de la posición de cada descriptor en los ejes X, Y, y Z. De esta manera, se genera una nube de puntos en la que cada punto es definido por una matriz que representa su ubicación en X, Y, y Z, así como los valores correspondientes al color de su descriptor como proporciones de rojo, verde, y azul (RGB).

Open Drone Map cuenta además con una función que emplea los datos de GPS contenidos en las imágenes para realizar un mapa tridimensional con referencias geográficas, lo que significa que la malla proyectada se encontrará a una escala y orientación acertada respecto al mundo real, lo que resulta especialmente útil para los motivos de este proyecto, pues si se tiene una escala acertada, es posible realizar mediciones de la región directamente sobre el mapa generado.

Para un funcionamiento más sencillo, ODM requiere que exista una carpeta dedicada a los proyectos de mapeo que este realizará, la cual es definida como parte del proceso de

instalación, indicado en la página oficial, dentro del archivo settings.yaml que se genera dentro de la carpeta de instalación de Open Drone Map. Esto permite que el sistema sea activado con solo especificar el nombre del paquete donde se encuentran las imágenes cuando se envía la instrucción en la terminal, sin tener que especificar la ruta completa. Cabe resaltar que para que esto funcione, dentro del paquete que representa a cada proyecto, debe existir una carpeta nombrada específicamente “images”, en la que se ubican las imágenes que serán procesadas.

```
# Uncomment lines as needed to edit default settings.
# Note this only works for settings with default values. Some commands like --rerun <module>
# or --force-ccd n will have to be set in the command line (if you need to)

# This line is really important to set up properly
project_path: '/home/user/Mapping'
# Example: '/home/user/ODMProjects
```

Figura 6: Fragmento del documento settings.yaml, donde se observa cómo se define la dirección donde se leerán y escribirán los archivos generados por ODM.

3.1.2: Open Source Computer Vision Library

Open Source Computer Vision Library (OpenCV) es una librería enfocada al procesamiento de imagen computacional lanzado bajo licencia BSD, lo que permite que sea usado de forma gratuita tanto para fines académicos como comerciales, y cuenta con interfaces de C++, Python, y Java con soporte para Windows, Linux, Mac OS, iOS, y Android, lo que implica una amplia gama de aplicaciones sobre las que puede ser utilizado.

Las especificaciones para su funcionamiento dependen del tipo de uso que al que se va a enfocar; si va a ser utilizado para procesar imágenes almacenadas, basta con especificar la dirección del archivo sobre el que se desea trabajar, por otro lado, en caso de ser utilizado para procesar imágenes obtenidas en tiempo real a través de un sensor, es necesario indicar el puerto a través del que este será leído, siendo que en algunos casos, como en este proyecto, se debe utilizar una herramienta de “traducción” de formatos para que OpenCV sea capaz de leer y procesar la imagen recibida. En ambos casos, la librería es capaz de encargarse de

generar una ventana sobre la que se muestre ya sea el procesamiento realizado, o el que se está llevando a cabo en tiempo real para que el usuario pueda monitorear los resultados.

OpenCV cuenta con una amplia gama de ejemplos que se descargan durante el proceso de instalación, los cuales ayudan a comprender el uso de sus herramientas más básicas hasta las más complicadas, permitiendo que un usuario sin experiencia en procesamiento de imagen logre realizar procesamientos de imagen de complejidad media en poco tiempo, o bien, que se enfoque en aprender a utilizar una herramienta específica para emplearla en un sistema de desarrollo. Una muestra de estos ejemplos, se puede observar en la Figura 7, donde se generó una interfaz para que el usuario sea capaz de definir la tonalidad que el programa buscará dentro de la imagen recibida para detectar y resaltar el centro de figuras circulares del color definido.

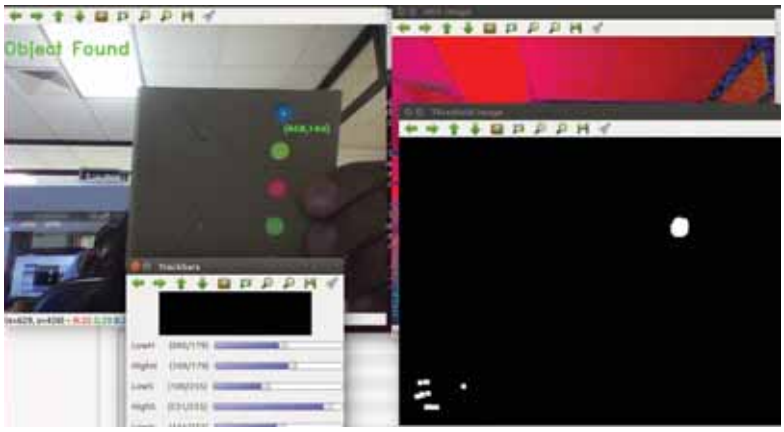

Figura 7: Detección de círculos de tonalidad variable en cámara web usando un código de ejemplo de OpenCV.

3.1.3: Point Cloud Library

Point Cloud Library (PCL) es un proyecto abierto a gran escala para procesamiento de imágenes y nubes de puntos en dos y tres dimensiones lanzado bajo licencia BSD, que contiene distintas aplicaciones y metodologías para algoritmos de filtrado, extracción de características, reconstrucción de superficies, segmentación de modelos, entre otros, que

pueden ser utilizados para filtrar datos de ruido, unir nubes de puntos tridimensionales, extraer puntos clave o encontrar descriptores correspondientes a un objeto en concreto, entre muchas otras aplicaciones.

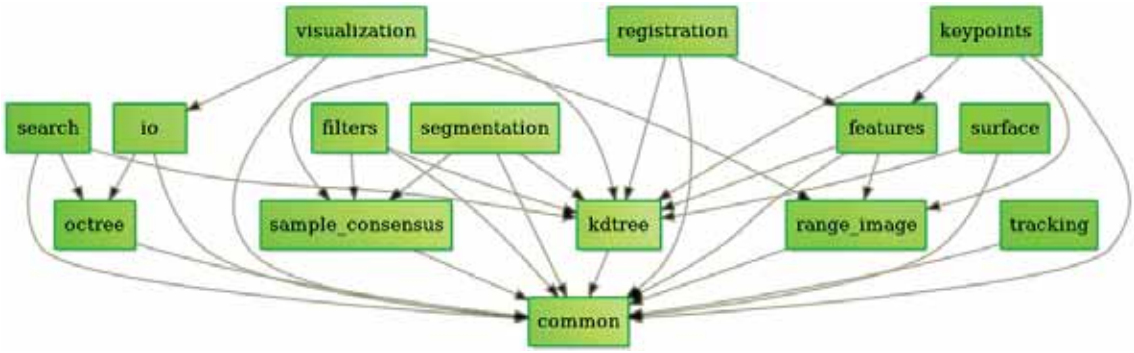

Figura 8: Gráfica de dependencia de Point Cloud Library (http://pointclouds.org/about/).

Es evidente que PCL resulta en una herramienta muy poderosa para el procesamiento de imágenes tridimensionales, sin embargo, y debido a su gran optimización sobre este aspecto, en el marco de este proyecto, Point Cloud Library fue explotado únicamente para una de sus aplicaciones más sencillas: la proyección de nubes de puntos ejecutada a través de la terminal. Esto último resultó en el factor determinante por el cual se usó esta herramienta en lugar de alguna otra, pues a diferencia de PCL, las demás no son capaces de realizar la proyección con al ejecutar una línea de comando desde la terminal.

3.1.4: Meshlab

Meshlab es un sistema de código abierto para el procesamiento y la edición de mallas tridimensionales, a través del cual es posible realizar diversas tareas de postprocesamiento tanto de nubes de puntos, como de mallas texturizadas. Realizado con el objetivo de ser una herramienta sustentable, eficiente, y fácil de utilizar, por lo que resulta en “una aplicación de visualización de mallas, donde un objeto tridimensional, almacenado en una variedad de formatos puede ser cargado e inspeccionado interactivamente de forma sencilla, simplemente

arrastrando y dando clic en la misma malla” (Cignoni et al., 2008).

Es por esta simplicidad, que se optó por utilizar esta herramienta, dado que el uso que se le dió a lo largo de este proyecto fue únicamente para visualizar e inspeccionar las mallas tridimensionales producidas por Open Drone Map, sin explotar sus herramientas de post procesamiento controlado de forma manual.

3.2: Algoritmo

El sistema está distribuido en tres etapas principales: selección de imágenes en tiempo real a partir de una transmisión de video, inicialización de mapeo, y visualización de resultados. Dependiendo del tipo de aplicación, el código desarrollado puede ser alterado a modo de ajustarse con los resultados esperados; en su formato original, los mapas son creados cada vez que se recupera un “n” número de imágenes, siendo “n” un parámetro ajustable. De este modo, el mapa visualizado cada vez que termina el proceso de mapeo actual representa la información de las imágenes captadas hasta el modelo previo junto con las nuevas imágenes recuperadas. Por otro lado, también es posible configurar el sistema para que el proceso de mapeo no inicie sino hasta que se ha finalizado por completo la etapa de filtrado de imágenes, es decir, hasta que se considera que durante el vuelo se ha captado información suficiente del área a mapear, de esta manera es posible decrementar el tiempo de procesamiento, pues se podría prescindir de las dos etapas consecuentes hasta el momento en que el vuelo ha finalizado, permitiendo así que, cuando se inicia el mapeo, todos los recursos del procesador sean dedicados exclusivamente a ello.

3.2.1: Filtrado de imágenes

El algoritmo de filtrado de imágenes está basado en el procedimiento Feature Matching de la librería de OpenCV (Open Source Computer Vision Library) (OpenCV library, 2018), que

analiza una imagen para encontrar puntos clave, denominados descriptores, los cuales son regularmente esquinas, dado que son los elementos más fáciles de detectar geométricamente. A través del método de comparación de fuerza bruta, el sistema extrae los descriptores de la primera imagen de referencia, la cual es obtenida inmediatamente al recibir imagen de video, es decir, en cuanto el sistema ha sido activado. Esta imagen es almacenada dentro de la base de datos para realizar los mapeos, y los descriptores que contiene son comparados con los que se encuentran procesando en tiempo real la señal de video recibida; de esta manera, una nueva imagen de referencia es obtenida y almacenada en la base de datos cuando la cantidad de puntos clave encontrados en la imagen en vivo, comparada con la referencia actual, es menor a un porcentaje de umbral definido en el código. Esto se hace con el objetivo de reducir la cantidad de imágenes que se analizan durante la siguiente etapa, lo que resulta en un decremento del tiempo total de procesamiento.

Es importante destacar que la imagen recibida por el dron a través de ROS se encuentra en un formato diferente al que OpenCV utiliza para realizar el procesamiento, por lo que se debe de utilizar una herramienta de ROS denominada CvBridge, la cual es capaz de traducir el mensaje de imagen recibido directamente del sensor de cámara del dron al formato cv::Mat que requiere OpenCV para realizar su funcionamiento. De esta manera, se generó un nodo que monitorea al tópico ‘/bebop/image_raw’ para buscar las imágenes que se espera sean publicadas por el dron, o por un publicador de video que emula el envío de mensajes del sensor fotográfico de un Bebop, lo que permite que los mensajes sean procesados en el código por la herramienta de CvBridge para convertirlos al formato en el que las herramientas de OpenCV pueden procesarlos. Además, durante este proceso, se crea una ventana dentro de la que se proyectarán las imágenes recibidas y la imagen de referencia.

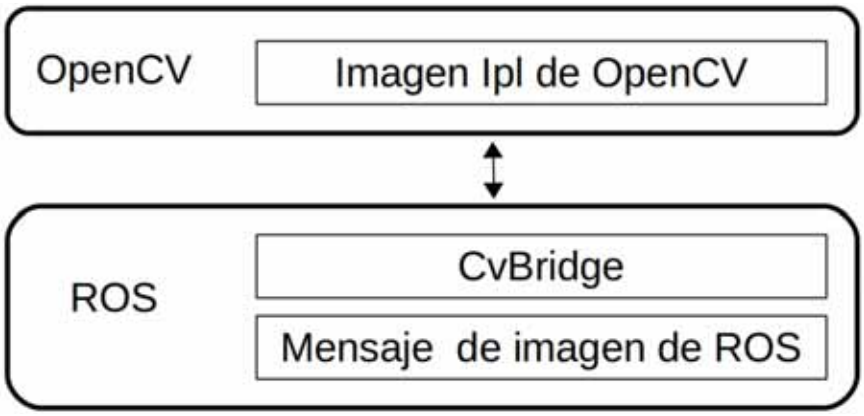

Figura 9: Representación gráfica del funcionamiento de ROS con OpenCV, a través de CvBridge.

En cuanto el video ha sido convertido al formato apropiado, se definen los arreglos que contienen a los descriptores tanto de la imagen en vivo, como de la imagen de referencia,al igual que la cantidad máxima de descriptores que se buscarán en cada imagen. Este último parámetro es especialmente importante, debido a que la cantidad de descriptores contenidos en la imagen determinan la eficacia del umbral de coincidencia que se usa para decidir el momento en el que se almacena un nuevo cuadro; una cantidad baja de descriptores provoca que el umbral sea vencido con facilidad, lo que resulta en un almacenamiento excesivo de imágenes, por lo que se encontró que una búsqueda de un máximo de 2200 descriptores es lo ideal, pues a partir de este umbral la cantidad de descriptores encontrados se mantiene prácticamente constante. Las Figuras 10 y 11 muestran la diferencia entre una búsqueda de 10 y 2200 descriptores, así como las Figuras 11 y 12 muestran la similitud en la cantidad de puntos encontrados entre una búsqueda de 2200 y 3000 descriptores.

Figura 10: Búsqueda de un máximo de 10 descriptores sobre la imagen.

Figura 11: Búsqueda de un máximo de 2200 descriptores sobre la imagen.

Figura 12: Búsqueda de un máximo de 3000 descriptores sobre la imagen.

Después de definir los elementos que contienen a los descriptores, es necesario definir los vectores que contienen tanto las posibles coincidencias como las coincidencias confirmadas, a partir de lo cual se inicia el procedimiento de comparación de descriptores de OpenCV, lo que implica la publicación de las imágenes dentro de la ventana creada previamente, una junto a la otra, sobre las cuales se dibujan círculos de múltiples colores encerrando a todos los descriptores encontrados en ambos lados de la ventana, los cuales son, a su vez, unidos por una línea con el descriptor que les corresponde sobre la imagen contraria a la que se encuentran ubicados, como se puede observar en las Figuras 13 y 14.

Figura 13: (Arriba) Muestra una comparación de imágenes con una búsqueda de 10 descriptores (se observa también que un mismo descriptor de la imagen almacenada fue unido erróneamente con dos descriptores de la imagen actual). (Abajo) Muestra una comparación de imágenes con una búsqueda de 100 descriptores.

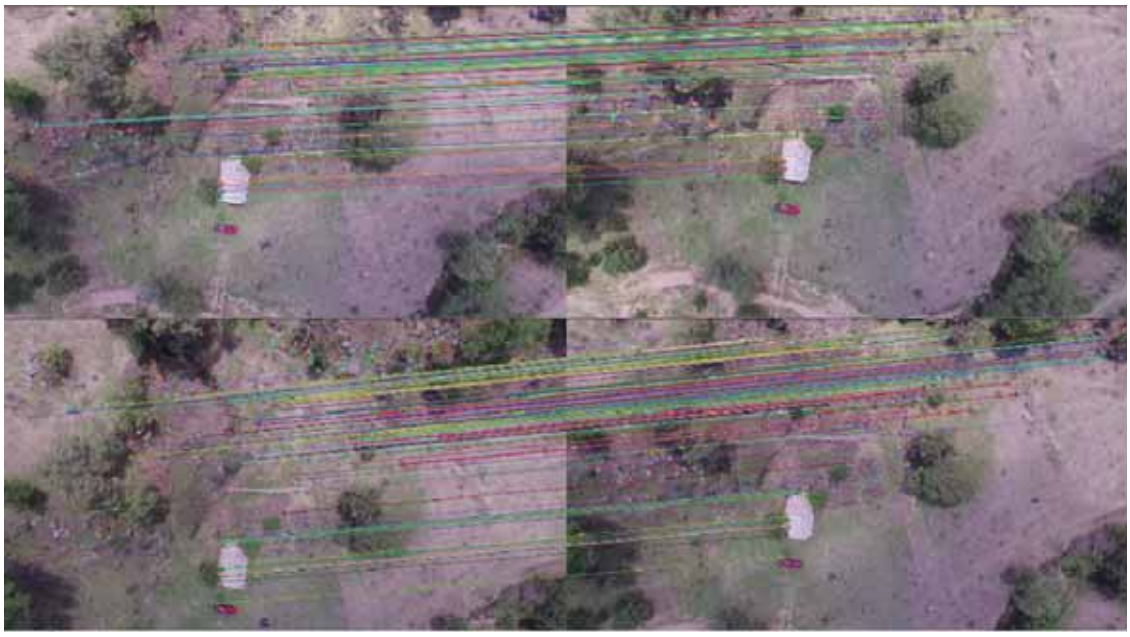

Figura 14: (Arriba) Muestra una comparación de imágenes con una búsqueda de 1000

descriptores. (Abajo) Muestra una comparación de imágenes con una búsqueda de 2200 descriptores, se observa una mayor cantidad tanto de descriptores dispersos, como de descriptores emparejados.

Finalmente, ya que el proceso de comparación de imágenes ha sido establecido, es necesario crear un procedimiento para garantizar que, una vez que el umbral ha sido superado, la imagen que se actualiza como la nueva imagen de referencia sea almacenada en el paquete indicado para realizar el mapeo. Para esto, dentro del código se especifican las instrucciones necesarias para crear uno o más nuevos directorios, según sea el formato de uso del sistema, y para crear dentro de estos el paquete “images” antes mencionado, en el que se deben almacenar las imágenes. Dado que el proceso se ejecuta desde la terminal, esto se logra utilizando usando la instrucción ‘mkdir’, que se encarga de crear paquetes en ubicaciones específicas con el nombre que se le indica; a su vez, para ejecutar la instrucción desde el código sobre la terminal de ejecución, esta es enviada usando la función ‘system()’, que toma como argumento una cadena que contiene la instrucción o serie de instrucciones que se desean ejecutar desde la terminal donde el código ha sido inicializado. La dirección sobre la que los paquetes serán creados, depende de la locación que el usuario haya seleccionado para la ubicación de los proyectos de ODM, en este caso, los paquetes deben ubicarse dentro del directorio “Mapping” en el folder personal, como se observa en la Figura 6.

3.2.2: Mapeo

La etapa de mapeo se realiza utilizando el sistema de postprocesamiento OpenDroneMap que, como se mencionó previamente, es capaz de entregar mapas tridimensionales texturizados, sin embargo, a modo de reducir el tiempo de procesamiento, en la mayoría de los casos se optó por detener la computación en cuanto se genera la nube de puntos; no

obstante, es posible permitir al sistema realizar el procesamiento completo de ser necesario.

En caso de que se desee que esta etapa se realice de manera paralela con el proceso de filtrado de imágenes, es decir, que se produzcan mapas cada cierto tiempo (medido en base a la cantidad de imágenes capturadas) durante el vuelo, es necesario que el mapeo sea inicializado en una terminal diferente de la que está llevando a cabo la primera etapa, pues de otra manera esta sería interrumpida hasta que el mapeo haya finalizado, lo que implica la pérdida de todas las imágenes que serían captadas durante este proceso. Para lograr esto, se hace uso nuevamente de la función 'system()', esta vez usando como primera parte del argumento la instrucción "gnome-terminal -x sh -c ' ' ", que indica la creación de una terminal externa, sobre la que es posible ejecutar nuevas instrucciones (colocadas dentro de los apóstrofes, y separadas por un punto y coma para indicar una instrucción diferente), por lo que es a partir de ella que se inicializa el proceso de Open Drone Map, con la posibilidad de realizar el mapeo completo, o bien, usar alguno de los parámetros de ejecución con los que cuenta; por ejemplo '--end-with ' seguido del paquete generado por ODM con el que se desea marcar el final del procesamiento (se usa 'opensfm' para detener el proceso cuando la nube de puntos ha sido generada).

Figura 15: Paquetes creados a partir del procesamiento de ODM; la nube de puntos se encuentra en el directorio /opensfm/depthmaps/merged.ply, mientras que la malla se encuentra en /odm_texturing/odm_textured_model.obj.

Además, antes de inicializar el mapeo, se accede a otra terminal para tomar algunas de las imágenes del paquete actual y copiarlas hacia el siguiente utilizando el comando ‘cp -b’, seguido de dos argumentos que representan el paquete donde existe el archivo, y el paquete hacia el que será copiado. Esto es realizado para garantizar que existan puntos coincidentes entre cada mapa y su posterior, de modo que se abra la posibilidad de computar una unión de nubes de puntos o mapas texturizados en post procesamiento.

3.2.3: Visualización

Finalmente, para la etapa de visualización de resultados, se generaron dos archivos ejecutables, uno para la proyección de nube de puntos, y uno para la corrección del archivo que los contiene, pues se encontró que para que la nube pueda ser proyectada, el archivo “merged.ply”, generado por Open Drone Map en el paquete “opensfm/depthmaps” dentro del paquete del proyecto, debe ser abierto y guardado, pues de otro modo el visualizador arroja un error de formato.

Para realizar estas dos acciones, se crearon archivos ejecutables a partir de un código en C++, el primero simplemente accede al archivo especificando la ruta completa de la ubicación del archivo de formato poligonal como argumento al ejecutar el archivo. El segundo es un corto código que hace uso de la librería conocida como Point Cloud Library para activar un visualizador sobre el que se proyecta la nube de puntos contenida en el archivo PLY afectado por el archivo ejecutable previo. Ambos archivos son ejecutados desde la terminal externa de la misma manera que se ejecuta ODM desde el código.

Es recomendable que estos archivos ejecutables sean colocados dentro del paquete de Open Drone Map, pues por simplicidad es desde este directorio que se comienza todo el sistema, debido a que el proceso de mapeo debe ser iniciado específicamente desde ese directorio, por lo que, al incluir los ejecutables de guardado y lectura de archivos poligonales, se reduce la

cantidad de instrucciones enviadas sobre la terminal externa, pues de este modo no es necesario realizar cambios de directorio. Además, se destaca que el proceso de visualización dentro del código funciona únicamente para nubes de puntos, si se desea visualizar el mapa tridimensional texturizado, se utiliza el visualizador de Meshlab, posterior al procesamiento.

Al ser un sistema basado en tres etapas, en la que las dos primeras son bastante robustas, el sistema puede resultar pesado para algunos equipos; especialmente si el control remoto del UAV se realiza a través de la misma computadora, lo que podría resultar en la pérdida de control del vehículo. Por esto, es necesario hacer énfasis en que es posible reducir las demandas de procesamiento con algunos simples ajustes en el código. Por ejemplo, de ser necesario reducir al máximo el consumo de recursos computacionales para garantizar el funcionamiento estable del equipo, el código puede simplificarse hasta contar únicamente con el filtrado de imágenes, de modo que, durante el vuelo, el sistema se dedique únicamente a recopilar las imágenes que caen bajo el umbral cuando son comparadas con la imagen previa; con esto, el proceso de mapeo puede ser realizado fuera de línea en cualquier momento haciendo uso de la base de datos que se generó con la primera etapa. En cuanto al consumo de recursos gráficos, el único punto del sistema que demanda esta clase de recursos es la visualización de resultados, de modo que esta etapa puede ser omitida por completo si se desea que el mapa o la nube de puntos generada sean visualizados en algún momento posterior al procesamiento. Por otra parte, la cantidad de mapas que se generan durante el vuelo puede verse reducida al incrementar la cantidad de cuadros que son capturados por paquete o reduciendo aún más el umbral de selección de imágenes (esto último puede afectar directamente la resolución y precisión del mapa tridimensional), provocando que el cambio de paquete se prolongue, lo que resulta en un retraso en el inicio del procesamiento del siguiente mapa actualizado.

Capítulo 4: Open Drone Map contra Agisoft

Para realizar con una comparación objetiva de los sistemas de mapeo tridimensional utilizados, se comenzó por realizar un mapeo en ambos sistemas, utilizando las mismas imágenes de entrada para observar la calidad del producto final entregado por cada uno, para esto, se tomaron en cuenta los elementos de tiempo de procesamiento para la generación de nube de puntos y de malla tridimensional, así como la cantidad de puntos generados. Estos aspectos fueron elegidos dado que, a partir de ellos, se puede hacer una estimación de la calidad computacional que estos producen; del mismo modo, se tomó en cuenta la calidad visual que ambos sistemas ofrecen en su producto final. Dado que la utilidad de cada uno de los sistemas para este proyecto es específicamente la generación de un mapa tridimensional, cualquier herramienta extra contenida en cualquiera de ellos para el análisis o post procesamiento de la malla es despreciada para los motivos de la comparación presentada a continuación.

El análisis muestra los resultados obtenidos usando cada sistema; cabe mencionar que el sistema de Open Drone Map ejecuta el procedimiento completo de manera automática desde que se le indica la ubicación del paquete que contiene su base de datos, mientras que en Agisoft es necesario introducir manualmente las imágenes a mapear, lo que permite que las imágenes seleccionadas sean adquiridas de un paquete de archivos sin filtrar, es decir, pueden ser tomadas desde cualquier carpeta sin importar cualquier archivo adicional que exista junto con ellas, mientras que ODM requiere de un paquete dedicado específicamente al proyecto de mapeo que incluya únicamente las imágenes que se usarán para realizar el mapeo. Del mismo modo, al usar Agisoft, es necesario comenzar manualmente los procesos de estimación de nube de puntos, generación de nube densa de puntos y generación de malla sobre la nube densa de puntos cada vez que el proceso previo ha concluido, por lo que es necesario monitorear el avance del proyecto para garantizar que este sea finalizado en el menor tiempo

posible, aún cuando no se desea modificar ninguna de las características predeterminadas durante dicho proceso.

Se destaca que el tiempo medido y presentado corresponde a un ordenador relativamente antiguo (fabricado en el año 2011) con procesador Intel i3; este aspecto afecta directamente al sistema de Open Drone Map, pues se ha observado que ordenadores más recientes y con mayor capacidad son capaces de reducir el tiempo de procesamiento de ODM hasta en un 80%.

La primera prueba fue realizada utilizando una base de datos que consta de cuatro imágenes con un índice de coincidencia de 40% entre ellas, mostradas en la Figura 16.

Figura 16: Imágenes utilizadas para computar el primer mapeo, es posible observar el porcentaje de coincidencia entre cada imagen y su posterior.

4 imágenes	Tiempo para generar una nube densa de puntos	Cantidad de puntos en la nube	Tiempo para generar un mapa tridimensional
Agisoft	11 s.	7,656	17 s.
ODM	3 min. 56 s.	51,799	5 min. 26 s.

Tabla 1: Comparación de aspectos clave en el procesamiento realizado por cada uno de los sistemas al realizar un mapa tridimensional a partir de cuatro imágenes.

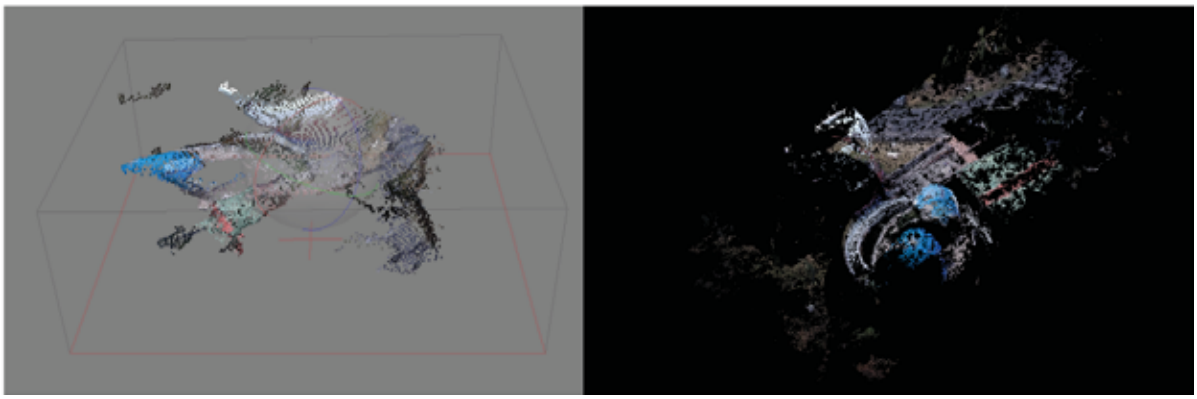

Figura 17: Comparación de las nubes de puntos generadas con cuatro imágenes en Agisoft (izquierda) y ODM (derecha).

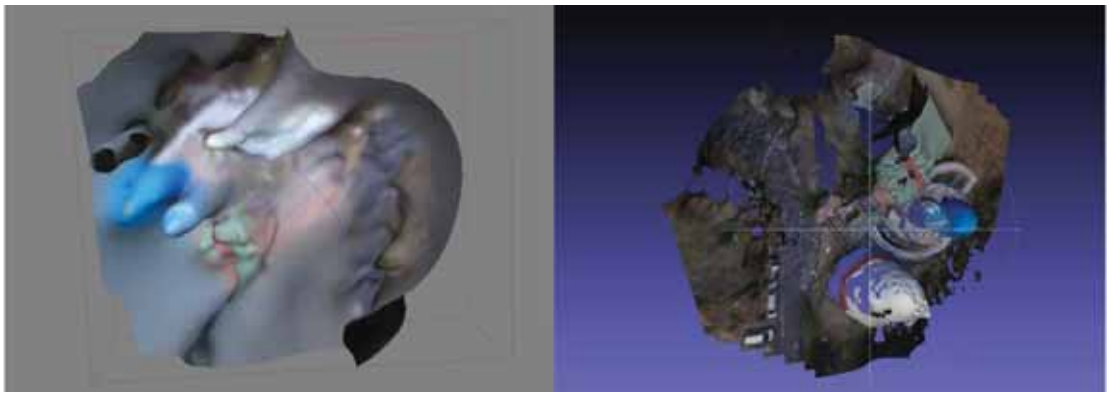

Figura 18: Comparación de las mallas tridimensionales generadas con cuatro imágenes en Agisoft (izquierda) y ODM (derecha).

Tras realizar la primera iteración, los resultados de la Tabla 1 demuestran que respecto al tiempo de procesamiento, el que ofrece que ofrece Agisoft es bastante superior, pues entrega un producto finalizado en casi 20 veces menos tiempo que ODM; sin embargo, en cuanto a la cantidad de puntos obtenidos, y calidad de la malla tridimensional generada, es evidente en las Figuras 17 y 18 que los resultados de Open Drone Map son superiores a pesar de la amplia brecha que existe en el tiempo de procesamiento, pues la cantidad de puntos obtenidos es 6.7 veces mayor usando ODM comparados con los de Agisoft, lo que implica una calidad considerablemente mejor.

La segunda prueba de calidad se realizó con diez imágenes seleccionadas manualmente de una base de datos más amplia, por lo que el índice de coincidencia varía en cada imagen contra su posterior con una diferencia subjetiva. Algunas de las imágenes se observan en la Figura 19.

Figura 19: Imágenes utilizadas para realizar el segundo proceso de mapeo.

10 imágenes	Tiempo para generar una nube densa de puntos	Cantidad de puntos en la nube	Tiempo para generar un mapa tridimensional
Agisoft	54 s.	510,051	2 min. 05 s.
ODM	10 min. 24 s.	211,045	13 min. 23 s.

Tabla 2: Comparación de aspectos clave en el procesamiento realizado por cada uno de los sistemas al realizar un mapa tridimensional a partir de diez imágenes.

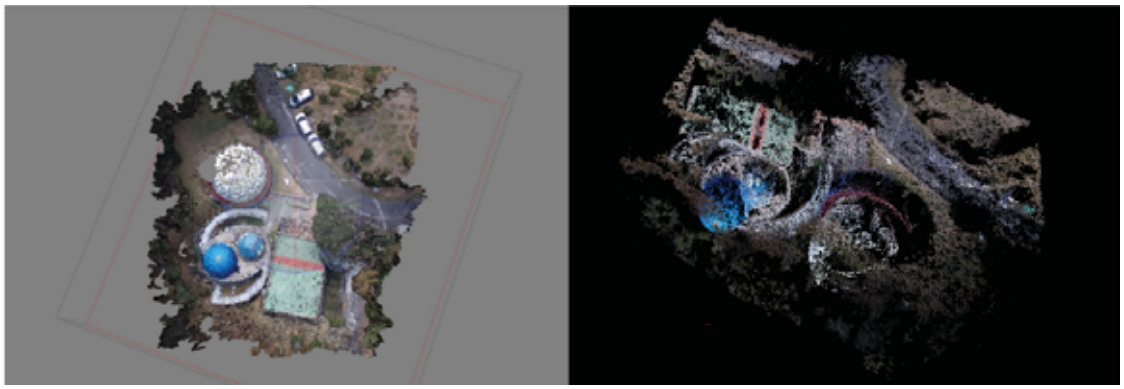

Figura 20: Comparación de las nubes de puntos generadas con diez imágenes en Agisoft (izquierda) y ODM (derecha).

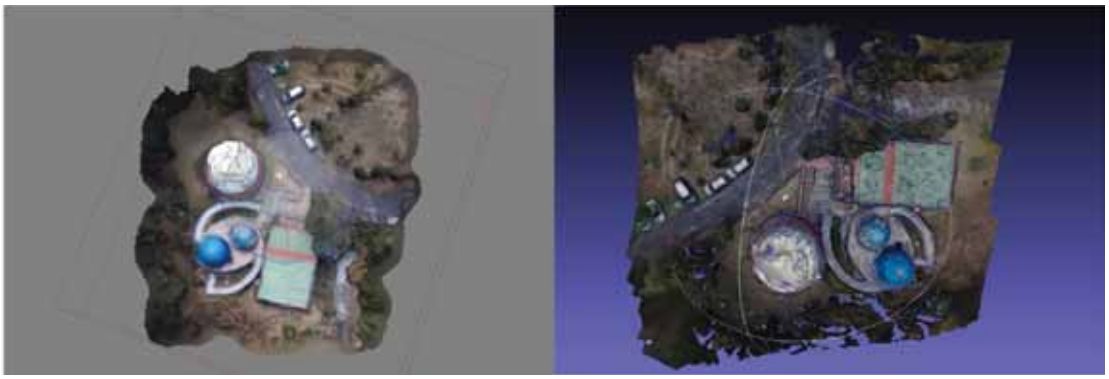

Figura 21: Comparación de las mallas tridimensionales generadas con diez imágenes en Agisoft (izquierda) y ODM (derecha).

En este caso, se observó que la computación de Agisoft entregó mejores resultados tanto en tiempo de procesamiento como en cuanto a la cantidad de puntos mapeados (Tabla 2), sin embargo, la malla finalizada que se muestra en la Figura 21 es de una calidad visual semejante entre ambas, con la diferencia de que Agisoft crea una malla cerrada, es decir, el sistema intenta rellenar cualquier hueco presente dentro de la frontera de puntos, lo que puede provocar pérdida de puntos ubicados más allá de la frontera mejor definida. Por otro lado, Open Drone Map trata de crear una malla utilizando todos los puntos existentes en la nube, por lo que la frontera cubre puntos más lejanos, con la consecuencia de generar algunos huecos en la malla.

Finalmente, se realizó una prueba usando 45 imágenes con un 60% de coincidencia, en la que se incluye un fragmento de bosque para comprobar la eficacia de cada sistema en este tipo de ambientes, donde la selección de puntos coincidentes es muy compleja debido a la similitud que hay en el ambiente a mapear, sin mencionar que, debido al viento, los árboles tienden a cambiar de posición de cuadro a cuadro.

45 imágenes	Tiempo para generar una nube densa de puntos	Cantidad de puntos en la nube	Tiempo para generar un mapa tridimensional
Agisoft	11 min. 17 s.	1,111,707	14 min. 39 s.
ODM	1 h. 11 min. 29 s.	1,004,556	1 h. 15 min. 52 s.

Tabla 3: Comparación de aspectos clave en el procesamiento realizado por cada uno de los sistemas al realizar un mapa tridimensional a partir de 45 imágenes.

Figura 22: Comparación de las nubes de puntos generadas con 45 imágenes en Agisoft (izquierda) y ODM (derecha).

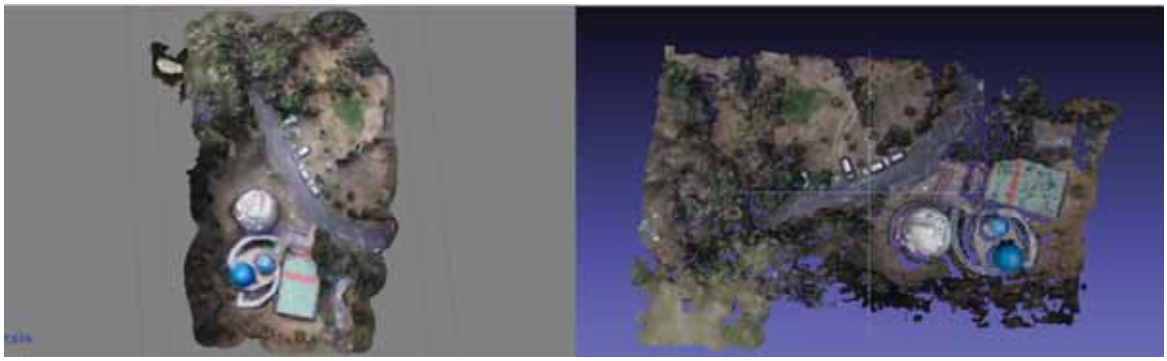

Figura 23: Comparación de las mallas tridimensionales generadas con 45 imágenes en Agisoft (izquierda) y ODM (derecha).

Al completar el mapeo de una base de datos tan amplia, se observa en la Tabla 3 que Open Drone Map toma un tiempo excesivo para terminar el procesamiento, además de que la cantidad de puntos incluidos en la nube es inferior a la que genera Agisoft con alrededor de una hora de ventaja, a pesar de lo cual, es observable en la Figura 23 que la malla de ODM produce una mejor representación de elementos aislados o de difícil texturización, como lo son los vehículos estacionados de este ejemplo, o lo árboles sobre los que voló el dron.

Finalmente, a pesar de que en ODM el tiempo de procesamiento incrementa ampliamente al incluir más imágenes para mapear, se destaca que este no requiere de una gran cantidad de

imágenes aéreas para entregar un mapa de calidad suficiente para realizar un análisis del terreno, lo que permite la reducción del umbral de coincidencia entre las imágenes que se capturan durante el vuelo. A partir de este análisis, se estima que el porcentaje de coincidencia ideal para usar el sistema es de aproximadamente 40% para evitar el uso de imágenes redundantes y tener la mayor eficiencia en cuanto al tiempo de procesamiento (esto en el caso de realizar mapas por etapas).

Capítulo 5: Desarrollo experimental y resultados

Se realizó un vuelo en la zona arqueológica de Tlalancaleca, ubicada en el municipio de San Matías Tlalancaleca, Puebla, en la que se voló un dron tipo DJI Matriz 100. Debido a limitaciones eléctricas, el sistema no fue utilizado directamente sobre el vuelo, sino que los archivos de video de dos secuencias aéreas fueron capturadas con la cámara del dron apuntando directamente hacia abajo, mientras el vehículo era piloteado para seguir una trayectoria sobre un área especificada por el Dr. Kabata. Esto permitió que se realizaran múltiples pruebas del sistema respecto al mismo vuelo, usando diferentes umbrales para la selección de imágenes.

Figura 24: Zona arqueológica de Tlalancaleca, Puebla.

Se ejecutó el sistema de filtrado de imágenes sobre uno de los videos capturados en el vuelo, utilizando diferentes umbrales para observar la cantidad de imágenes producidas, lo que permitiría fijar el umbral ideal para situaciones en las que el sistema se use directamente

sobre el vuelo para facilitar el análisis de datos sin tener que dejar la zona arqueológica, de modo que sea posible determinar si es necesario realizar un nuevo vuelo sobre la misma área, o si la información recopilada es suficiente para avanzar a otra región de la zona. El proceso se llevó a cabo dos veces para verificar la objetividad del sistema, la cual fue comprobada con los resultados observados en la Tabla 4.

Porcentaje de coincidencia	Cantidad de imágenes filtradas en la primera iteración	Cantidad de imágenes filtradas en la segunda iteración
5%	60	59
10%	77	77
15%	87	89
20%	101	102

Tabla 4: Imágenes obtenidas a través del algoritmo de filtrado con diferente porcentaje de coincidencia requerido.

A partir de estos resultados, se realizaron los mapeos correspondientes a cada umbral a modo de observar las brechas de calidad en cuanto a las mallas que se generan en cada caso.

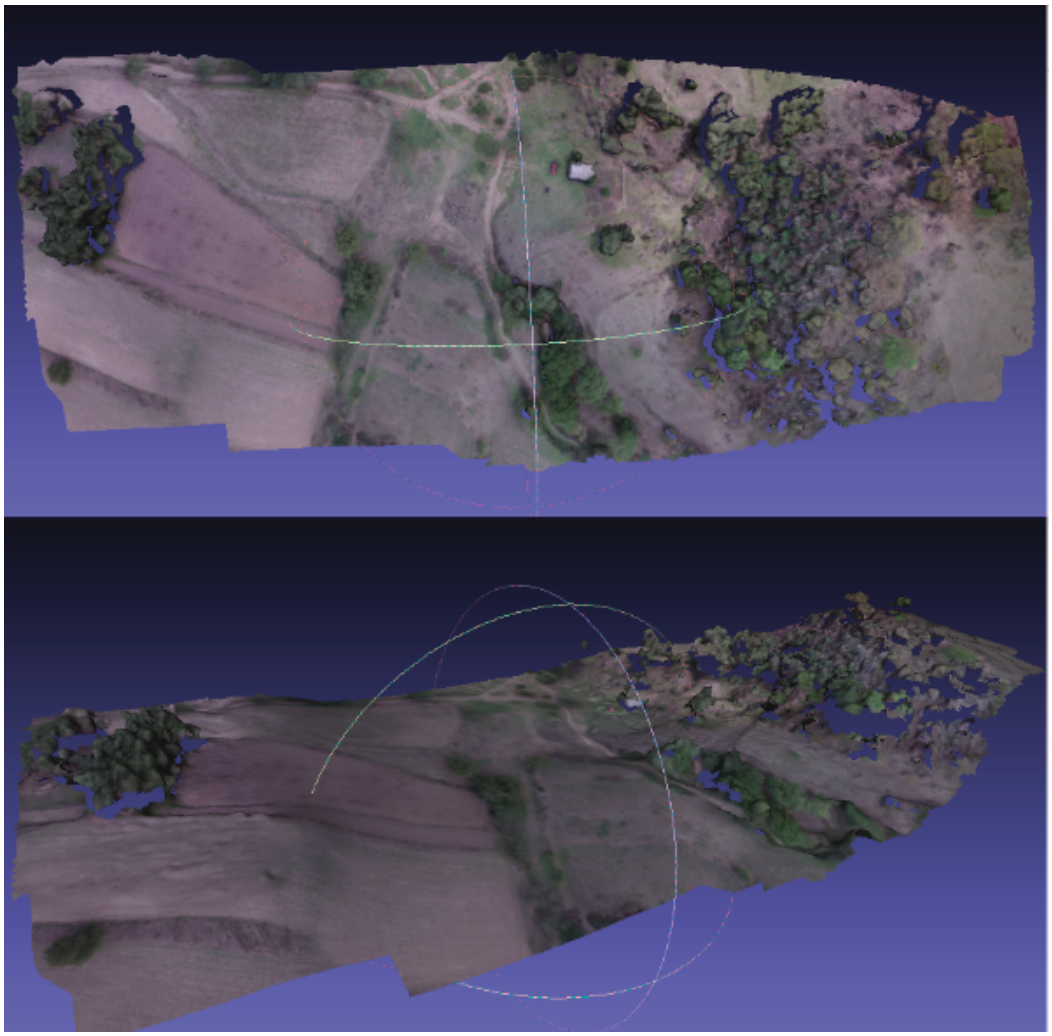

Figura 25: Vista superior (arriba) y panorámica (abajo) del mapa tridimensional generado con imágenes al 5% de coincidencia.

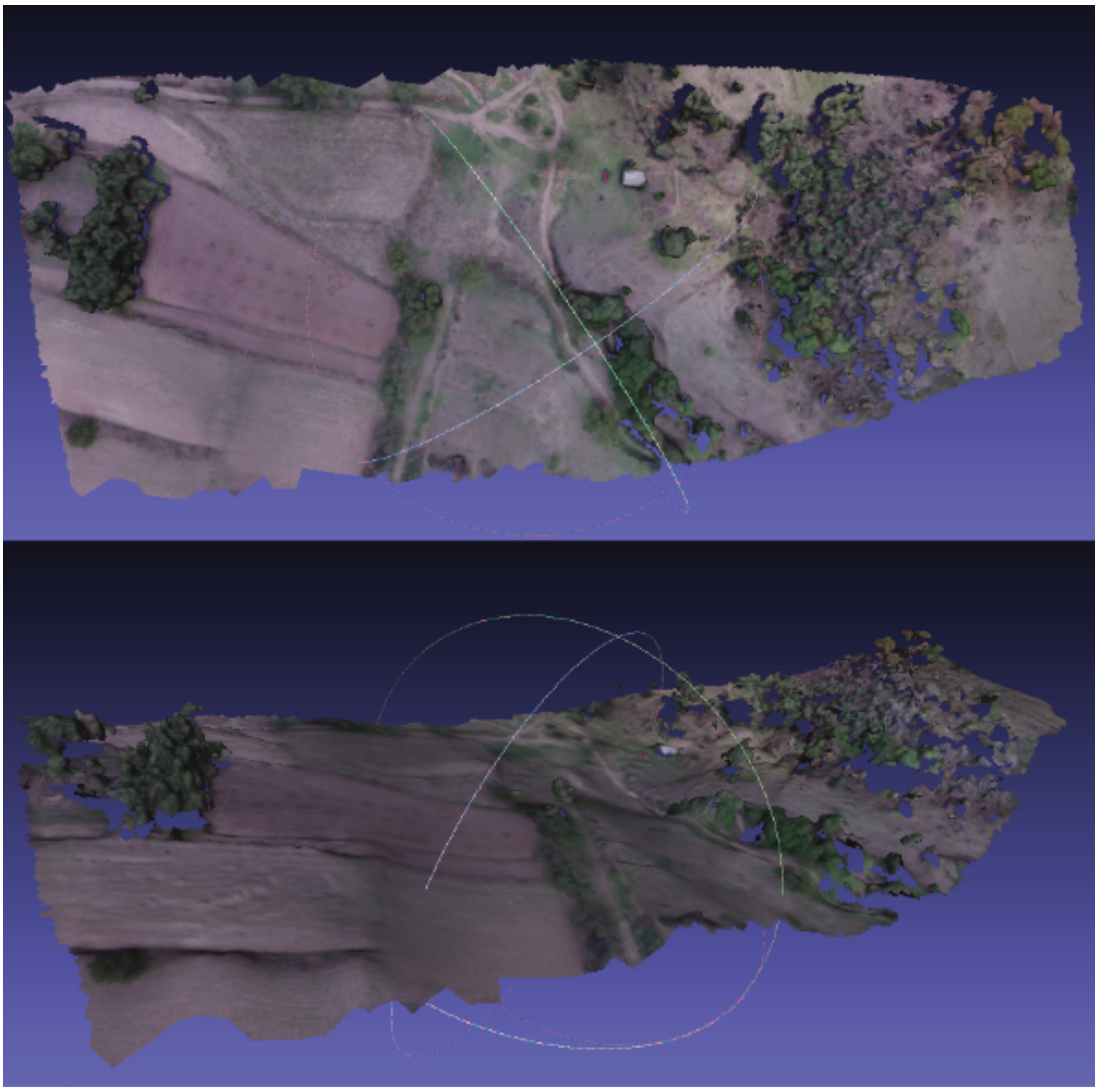

Figura 26: Vista superior (arriba) y panorámica (abajo) del mapa tridimensional generado con imágenes al 10% de coincidencia.

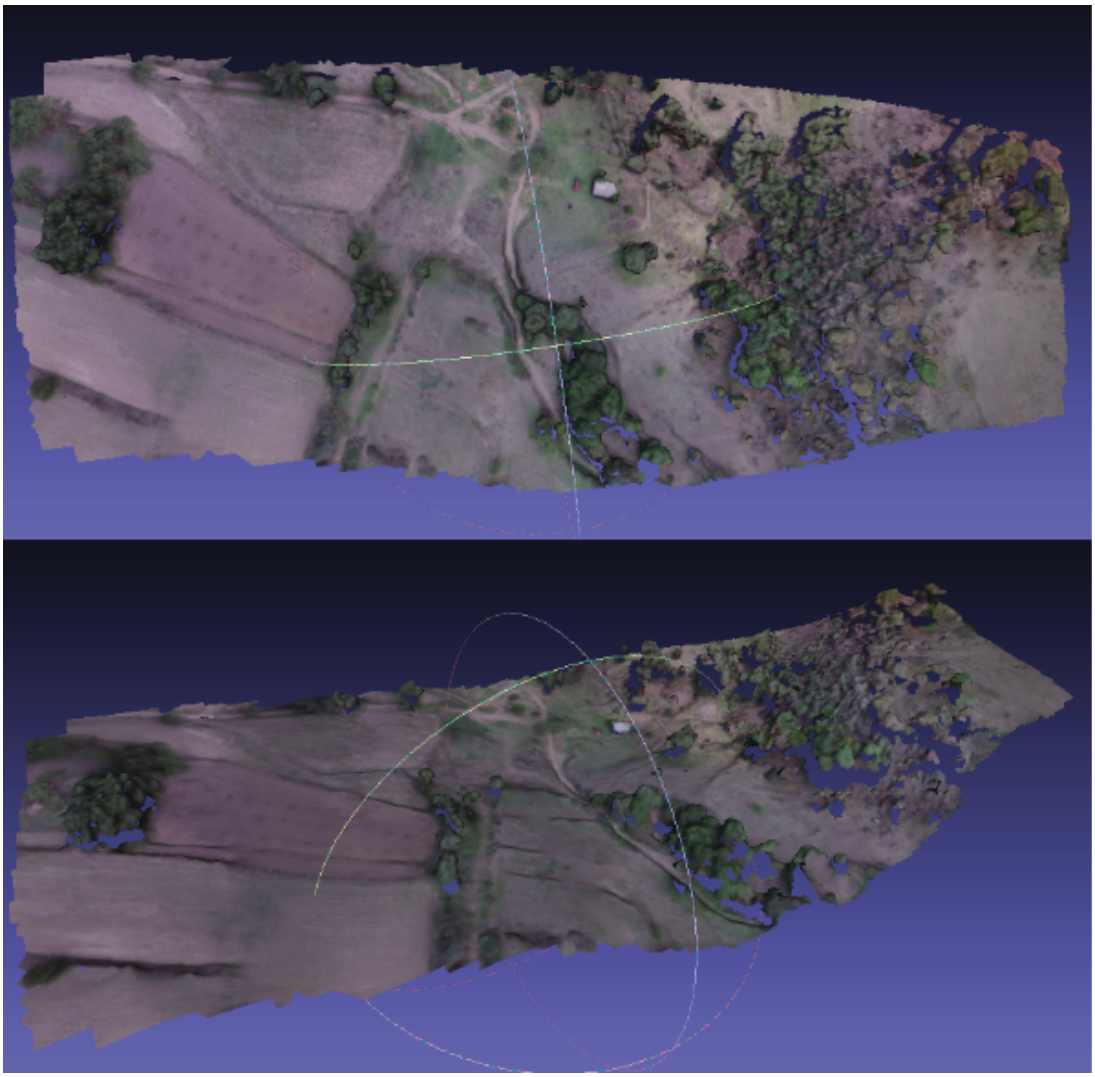

Figura 27: Vista superior (arriba) y panorámica (abajo) del mapa tridimensional generado con imágenes al 15% de coincidencia.

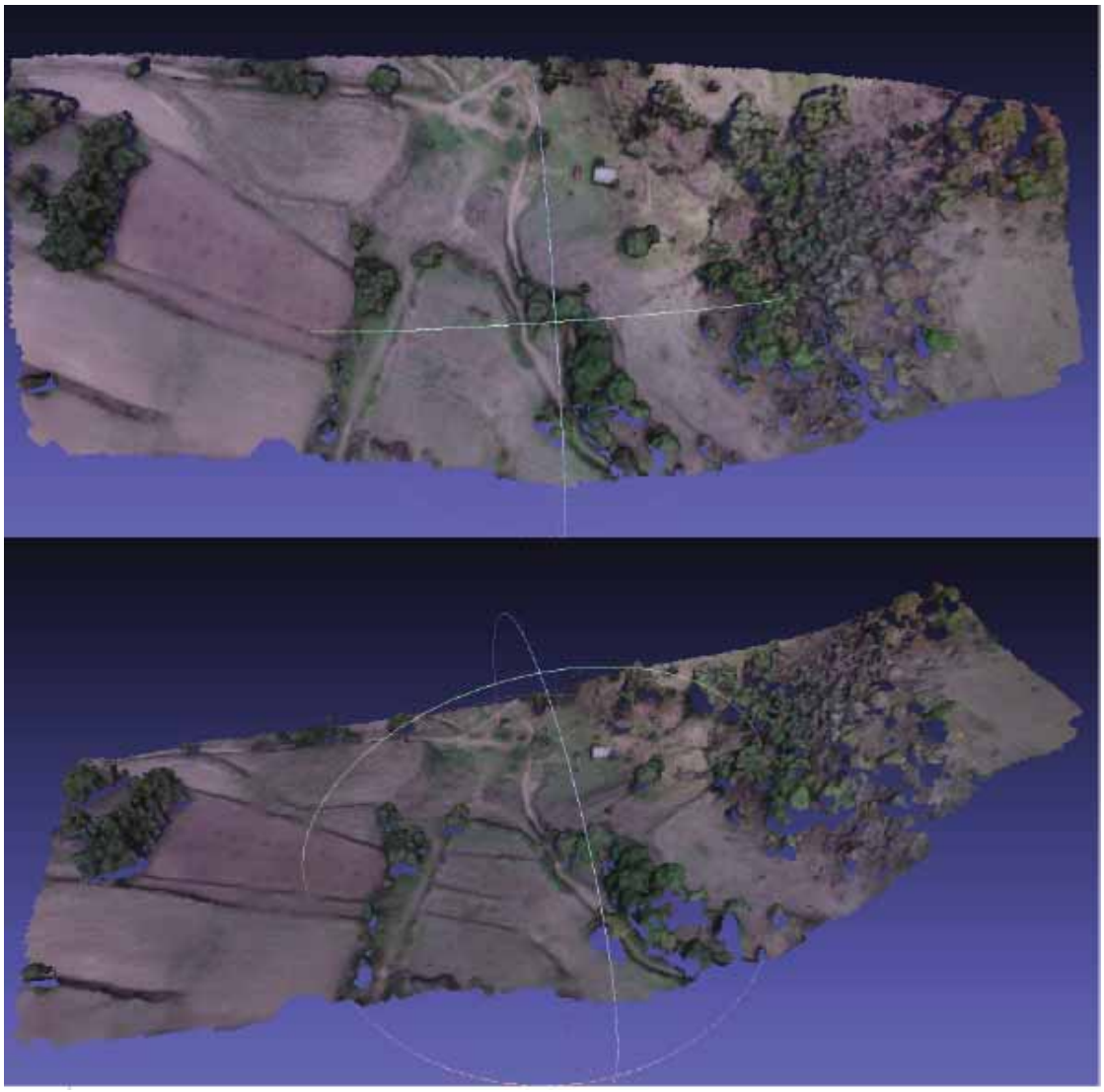

Figura 28: Vista superior (arriba) y panorámica (abajo) del mapa tridimensional generado con imágenes al 20% de coincidencia.

A pesar de que se observaron resultados similares en todos los casos, existe un nivel de semejanza más elevado especialmente entre los umbrales de 5% y 10% de 15% y 20%, lo cual es observable en las Figuras 25, 26, y 28, por lo que, considerando que en los segundos existen ciertos detalles que logran ser mapeados, y que no existen en los primeros, como se muestra en la Figura 29, se considera que un umbral de 15% es ideal para realizar mapeos a esta escala.

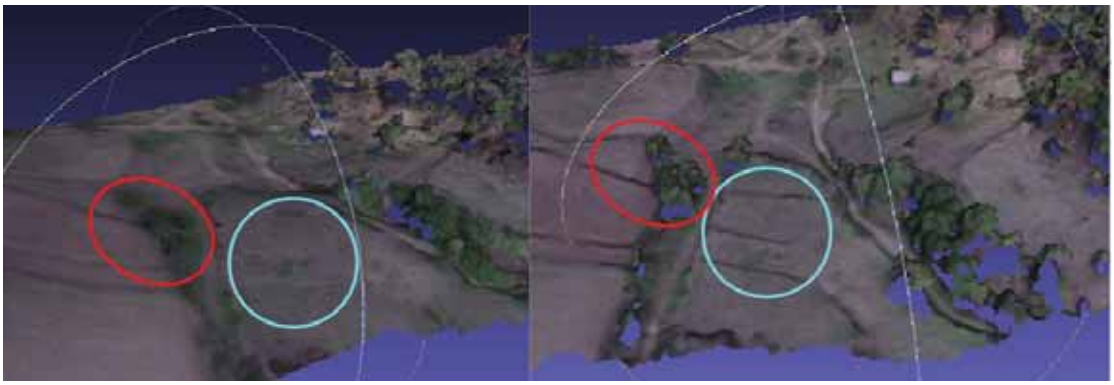

Figura 29: Comparación de elementos tridimensionales entre los mapas producidos con umbral de 5% (izquierda) y 20% (derecha).

En todos los mapas generados, se inspeccionó una de las regiones de principal interés para el Dr. Kabata, la cual está casi completamente cubierta por árboles, lo que resulta en una zona difícil para el procesamiento.

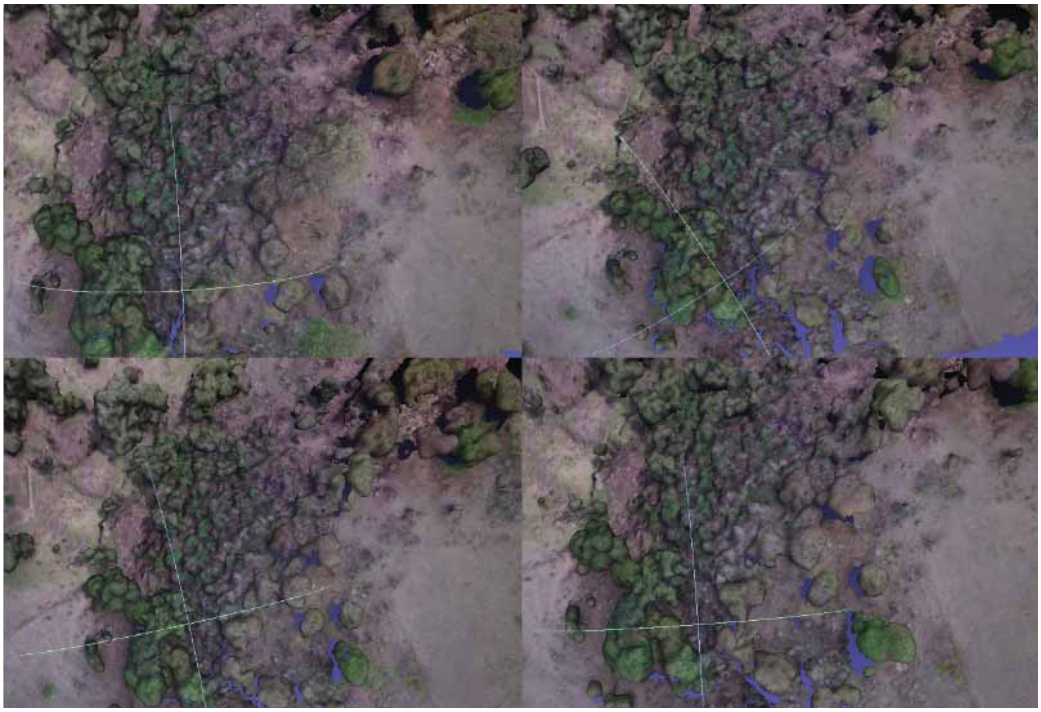

Figura 30: Visualización superior de zona de interés bajo los umbrales de 5% (arriba, izquierda), 10% (arriba, derecha), 15% (abajo, izquierda), y 20% (abajo, derecha).

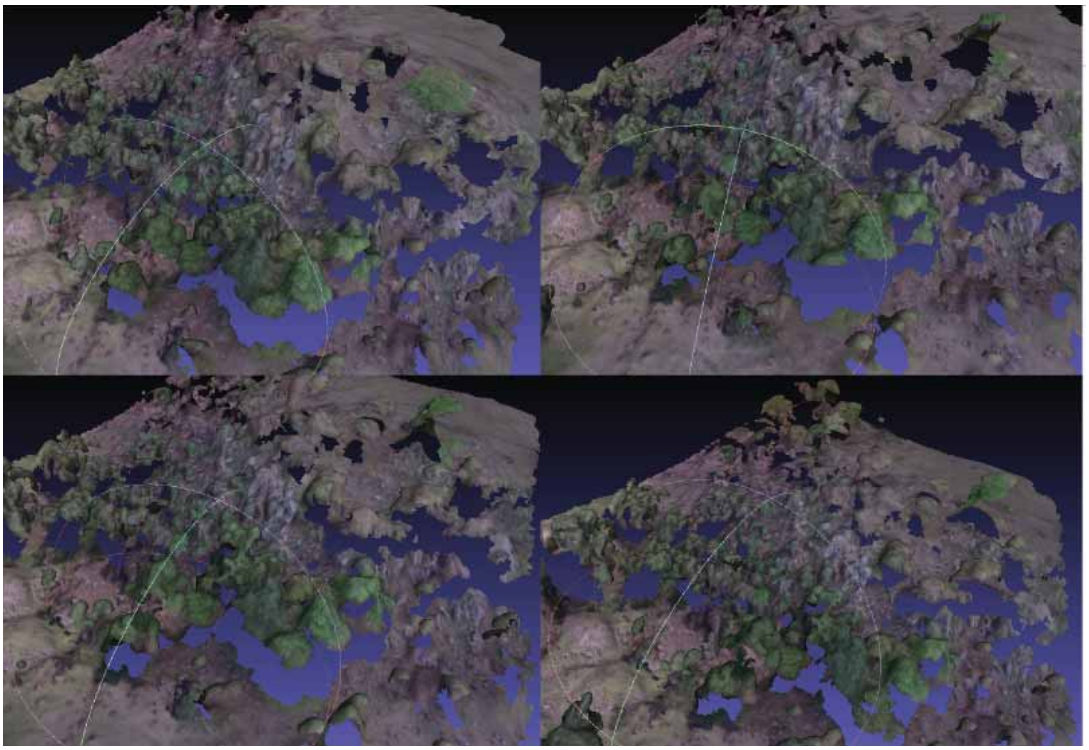

Figura 31: Visualización frontal de zona de interés bajo los umbrales de 5% (arriba, izquierda), 10% (arriba, derecha), 15% (abajo, izquierda), y 20% (abajo, derecha).

En todos los casos, la visualización vertical (Figura 30) resulta acertada y comparable con un resultado producido por Agisoft, mostrado en la Figura 32; sin embargo, al inspeccionar a detalle esta zona dentro de los mapas, como la sección mostrada en la Figura 31, es evidente que hace falta refinar la malla alrededor de esos puntos, lo cual podría ser fácilmente conseguido al capturar imágenes frontales y laterales de esta región en específico. Visualizar aspectos como este durante la jornada de mapeo, habría permitido la recopilación de datos adicionales que sirvieran para completar estos detalles, pues un vuelo de no más de un minuto con la cámara apuntando hacia los sectores donde existen huecos en el mapa sería suficiente para adquirir los datos necesarios para rellenarlos, o bien, determinar si es necesario realizar otro vuelo enfocado hacia puntos aún más específicos.

Figura 32: Vista superior de la nube de puntos generada con Agisoft usando un umbral de 15% de coincidencia entre cada imagen.

Capítulo 6: Conclusiones

Al utilizar el sistema sobre un video capturado del vuelo (que, para la forma en la que se implementó, equivale a utilizarlo durante el vuelo) se encontró una enorme reducción en cuanto al tiempo que toma tradicionalmente obtener las imágenes necesarias para realizar el mapeo, pues la primera etapa del sistema permite que estas sean adquiridas de forma automática al mismo tiempo que el vuelo está en curso, de modo que el operador del dron solo debe concentrarse en seguir una trayectoria; en contraste con realizar la captura de imágenes de forma manual, pues en este caso, el operador debe reubicar el enfoque de la cámara hasta asegurar, de manera subjetiva, que se encuentra en un punto donde el cuadro ha cambiado hasta contener un cierto porcentaje de coincidencia respecto a la imagen capturada previamente. Este proceso manual implica no solo un costo temporal, sino también uno energético, dado que este agota la batería del dron en momentos donde el dron no se mueve mientras se mantiene en el aire, o bien, realiza movimientos muy lentos para permitir al operador alinear adecuadamente el enfoque, lo que resultaba en el uso de múltiples baterías para realizar un solo proceso de mapeo, mientras que el uso de este sistema permite realizar los mapeos de hasta tres regiones dentro de la zona arqueológica con una sola carga, si el vuelo se realiza de forma eficiente, considerando que una batería completamente cargada es capaz de soportar entre 15 y 30 minutos de vuelo, dependiendo del dron utilizado.

Se observó también que una herramienta de mapeo de código libre, como Open Drone Map, a pesar de las brechas temporales que se incrementan conforme al incremento de la cantidad de imágenes usadas por mapa, es capaz de producir mapas tridimensionales de una calidad comparable con los creados a través de un sistema como Agisoft, que tiene un costo de $549USD por licencia, siendo que ODM captura una cantidad semejante de puntos para la nube (incluso superior, cuando se usa una baja cantidad de imágenes); lo que representa un gran decremento en cuanto al costo del proyecto, siendo el único detalle que, para operar

ODM, se requiere un nivel más avanzado experiencia en el ámbito computacional para utilizarlo, respecto al nivel requerido para utilizar Agisoft, pues este está diseñado para tener un uso intuitivo para el usuario, mientras que Open Drone Map está optimizado para funcionar a través de la terminal de Ubuntu 16.04 (versiones anteriores requieren el uso de Docker). Además, una gran ventaja con la que cuenta ODM para el estudio de zonas arqueológicas es la capacidad de producir mapas con escala precisa a metros si las imágenes utilizadas para el mapeo están georeferenciadas.

Capítulo 7: Trabajo futuro

El trabajo realizado durante este proyecto, así como las condiciones en las que este se encuentra, permiten pensar en la continuación de su desarrollo. Para empezar, se plantea la posibilidad de desarrollar un sistema en el que el mapeo sea realizado en vivo, y que permita al usuario observar los resultados mientras estos están siendo creados, lo que facilitaría aún más el trabajo de mapeo, pues, mientras el vehículo se encuentra en vuelo, el piloto podría observar si la ruta que está siguiendo es la más adecuada y determinar si debe corregir la ruta o regresar a algún punto específico para recopilar más datos.

Otra posibilidad es la de crear un sistema capaz de recibir la localización por GPS del UAV para permitir la selección de una ruta que se le indicará seguir para la obtención de datos que ingresarán al sistema de mapeo, reduciendo al mínimo la interacción humana con el sistema completo.

Finalmente, dentro del marco arqueológico, se plantea la posibilidad de implementar al sistema un método de detección y filtrado de ciertas características consideradas como ruido, así como de reconstrucción automática para las características importantes que las anteriores bloquean; lo que resultaría especialmente útil en casos como la región mapeada en este proyecto, pues hay ciertos elementos que se desean resaltar, los cuales están ocultos bajo diversos tipos de vegetación.

Bibliografía

- Micheletti N., Chandler J., & Lane S.. (2015). Structure from Motion (SfM) Photogrammetry. British Society for Geomorphology, Geomorphological Techniques, Chap. 2, Sec. 2.2.
- Westoby M, Brasington J, Glasser NF, Hambrey MJ, Reyonds MJ. 2012. Structure from Motion photogrammetry: a low-cost, effective tool for geoscience applications. Geomorphology 179: 300-314.
- Förstner, W., 1986. A feature-based correspondence algorithm for image matching. International Archives of Photogrammetry and Remote Sensing, 26: 150-166.
- Fergus, Rob. (2013). *'Lecture 6: Multi-View Stereo & Structure from Motion – 11_12_multiview.pdf'*<http://cs.nyu.edu/~fergus/teaching/vision/11_12_multiview.pdf >.
- P. Cignoni, M. Callieri, M. Corsini, M. Dellepiane, F. Ganovelli, & G. Ranzuglia. (2008). MeshLab: an Open-Source Mesh Processing Tool. En Sixth Eurographics Italian Chapter Conference(pp.129-136). Pisa, Italia: Visual Computing Lab, ISTI - CNR.
- Micheletti N, Chandler JH, Lane SN. 2014. Investigating the geomorphological potential of freely available and accessible structure-from-motion photogrammetry using a smartphone. Earth Surface Processes and Landforms, DOI: 10.1002/esp.3648.
- Thompson, EH. 1965. Review of methods independent model aerial triangulation. The Photogrammetric Record, 5:72–79.
- Brown, DC, 1971. Close-Range Camera Calibration. Photogrammetric Engineering, 37(8): 855–866.
- Brown, DC. 1976. The bundle adjustment-progress and prospects. International Archives of Photogrammetry. 21 B3, Helsinki.
- Granshaw SI. 1980. Bundle adjustment methods in engineering photogrammetry. The Photogrammetric Record10: 181-207.
- Faig W, Moniwa H. 1973. Convergent photos for close-range. Photogrammetric Engineering and Remote Sensing39: 605-610.
- Kenefick JF, Gyer MS, Harp BF. 1972. Analytical self-calibration. Photogrammetric Engineering and Remote Sensing38: 1117-1126.
- López, J., Kabata, S., Texis, A., & Fukahara H.. (2015, mayo 20). Mapeo y registro de artefactos en 3D utilizando Agisoft PhotoScan y Drone en el Proyecto Arqueológico Tlalancaleca, Puebla. El patrimonio cultural y las nuevas tecnologías, Segundo congreso internacional.
- Agisoft LLC. (2017). Agisoft PhotoScan User Manual: Professional Edition, Version 1.3. 2018, de Agisoft LLC Sitio web: http://www.agisoft.com/pdf/photoscan-pro_1_3_en.pdf.

- Ramirez, J. (1991). Computer-Aided Mapping Systems: The Next Generation. Center for Mapping, The Ohio State University.
- Krule, J.. (2014). The origins of aerial photography. 2018, de The Newyorker Sitio web: https://www.newyorker.com/culture/photo-booth/origins-aerial-photography.
- Staiger, R., 2003., Terrestrial Laser Scanning–Technology, System and Application, FIG Regional Conference Proceedings ISBN 87-90907-28-0, Marrakech, Morocco, December 2-5, 2003, Editors: Taïb Tachalait Dan J. Schnurr, TS 12.3– Positioning and Measurement Technologies and Practices, 2003.
- Lin, H. (2002). Computer Vision Techniques for Complete 3D Model Reconstruction (Tesis doctoral). State University of New York at Stony Brook, New York.
- Waechter M., Moehrle N., Goesele M. (2014) Let There Be Color! Large-Scale Texturing of 3D Reconstructions. In: Fleet D., Pajdla T., Schiele B., Tuytelaars T. (eds) Computer Vision – ECCV 2014. ECCV 2014. Lecture Notes in Computer Science, vol 8693. Springer, Cham.
- Zygmunt, M. (2013). THE TESTING OF PCL: AN OPEN-SOURCE LIBRARY FOR POINT CLOUD PROCESSING . Geomatics, Landmanagement and Landscape, No. 3, pp.105-115.
- Agisoft Photoscan. (2018). http://www.agisoft.com/.
- OpenCV Library. (2018). https://opencv.org/.
- Barnhart, E.. (2004). Proyecto de mapeo de Palenque. Reporte de la temporada de campo de 1998, FAMSI.